Mixed-variable Bayesian optimization : application to aerospace system design

Julien Pelamatti

Mixed-variable Bayesian optimization
Application to aerospace system design

A dissertation submitted by

Julien Pelamatti

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
(Informatics)

École Doctorale Sciences Pour l'Ingénieur
Université de Lille
Lille, France

&

ONERA – The French Aerospace Lab
Palaiseau, France

&

CNES – Direction des lanceurs
Paris, France

Defended publicly on the 9th of March 2020 in front of a defense committee composed of:

Dr. Mathieu BALESDENT	ONERA, Palaiseau	Co-advisor
Pr. Thomas BARTZ-BEIELSTEIN	TH Köln	Examiner
Dr. Loïc BREVAULT	ONERA, Palaiseau	Co-advisor
Pr. Rodolphe LE RICHE	CNRS at École des Mines de Saint-Étienne	Reviewer & Jury president
Dr. Amandine MARREL	CEA, Cadarache	Reviewer
Pr. El-Ghazali TALBI	INRIA Lille-Nord Europe	Thesis director
Yannick GUERIN	CNES, Direction des lanceurs	Invited

Optimisation Bayésienne de problèmes à variables mixtes
—
Application à la conception de véhicules spatiaux

Thèse de doctorat présentée et soutenue par

Julien Pelamatti

en vue d'obtenir le grade de

Docteur d'université
spécialité: informatique

École Doctorale Sciences Pour l'Ingénieur
Université de Lille
Lille, France

&

ONERA – The French Aerospace Lab
Palaiseau, France

&

CNES – Direction des lanceurs
Paris, France

Soutenue publiquement le 9 Mars 2020 à l'Office National d'Études et de Recherches Aérospatiales de Palaiseau devant un jury composé de:

Dr. Mathieu BALESDENT	ONERA, Palaiseau	Co-encadrant
Pr. Thomas BARTZ-BEIELSTEIN	TH Köln	Examinateur
Dr. Loïc BREVAULT	ONERA, Palaiseau	Co-encadrant
Pr. Rodolphe LE RICHE	CNRS à l'École des Mines de Saint-Étienne	Rapporteur & Président du jury
Dr. Amandine MARREL	CEA, Cadarache	Rapporteur
Pr. El-Ghazali TALBI	INRIA Lille-Nord Europe	Directeur de thèse
Yannick GUERIN	CNES, Direction des lanceurs	Invité

This manuscript is based on the *Masters / Doctoral Thesis* template (version 2.5).

BibTeX entry:

```
@PHDTHESIS{Pelamatti_PhD_2020,
author = {Pelamatti, Julien},
year = 2020,
title = {{Mixed-variable Bayesian optimization - Application to aerospace system design}},
school = {Universit\'e de Lille}
}
```

À Alice,

Remerciements / Acknowledgements

Me voici confronté au tout dernier et plus particulier exercice de cette rédaction, les remerciements. Ces trois années auront été une expérience de croissance aussi bien personnelle que professionnelle absolument immense, et même si j'aime bien voir cette thèse comme mon exploit, il y a de nombreuses personnes sans lesquelles cela aurait été nettement plus difficile et éprouvant, voir impossible, que je tiens donc à remercier. J'annonce tout de suite que ce genre de textes n'est clairement pas mon point fort, et si jamais je devais oublier quelqu'un je m'excuse, ce n'était (probablement) pas fait délibérément.

Je tiens tout d'abord à remercier mes deux encadrants, Loïc Brevault et Mathieu Balesdent, un binome hors-pair (héhé) sans lequel ce manuscrit n'existerait sûrement pas. Au long des hauts et des bas de cette thèse, ils ont toujours dédié une quantité de temps et d'énergie absolument énorme pour me suivre quotidiennement, pour me motiver et me pousser à toujours faire du travail rigoureux et de qualité, tout en me laissant l'autonomie pour faire tous les mauvais choix sans lesquels ça aurait risqué d'être une bonne thèse. On partait de loin, mais ils n'ont jamais arrêté de persévérer, et j'espère sincèrement qu'ils sont fiers du résultat final.

Je remercie aussi mon directeur de thèse El-Ghazali Talbi ainsi que Yannick Guerin, pour avoir accepté de suivre ma thèse et pour s'être assurés que je ne perde jamais le nord.

Je tiens bien évidemment à remercier Amandine Marrel et Rodolphe Le Riche qui ont accepté la lourde tâche d'être les rapporteurs de ce manuscrit. Leur retour extrêmement détaillé et méticuleux s'est révélé aussi utile que impressionnant.

Un remerciement particulièrement spécial va à mon co-bureau, Ali Hebbal. Tout d'abord pour sa compagnie au quotidien, pour sa bonne humeur constante, pour sa patience sans limites et pour nos conversations sur des sujets tout à fait variés, allant des jeux-vidéos à la géopolitique. Mais aussi pour m'avoir aidé et prêté ses connaissances mathématiques tout au long de la thèse, qui m'ont permis d'éviter plusieurs gaffes et de gagner énormément de temps. Tout ceci fait de lui, *de facto*, un co-auteur de cette thèse.

Bien évidemment, je remercie tous les stagiaires et les doctorants que j'ai eu le plaisir et l'opportunité de côtoyer pendant ces trois ans, pour la bonne ambiance pendant les pauses déjeuner ainsi que pour les sorties et activités diverses (même si on attend encore celle d'Ali). Je me vois tout de même obligé de faire quelques mentions spéciales aux personnes qui ont le plus marqué mes années de thèse. En ordre d'ancienneté, je commence par remercier Vincent Chabridon et Romain Rincé, mes premiers voisins de bureau, pour m'avoir tenu compagnie pendant la première moitié de la thèse, et m'avoir occupé bien au delà des pauses café avec des discussions aussi longues que passionnantes, même si parfois un peu frustrantes (#CapitainePelamatti). Vincent mérite aussi un seconde mention honorable pour avoir joué le rôle de grand frère académique dans la grande famille des thèses sur les lanceurs, en me conseillant dans certains moments difficiles et en s'intéressant à mon futur professionnel, même au coût de sa propre frustration.

Les prochains remerciements vont à Enzo Iglesis et Camille Palmier, héritiers de Romain et Vincent. Je remercie Enzo pour m'avoir obligé à surveiller de très près ma pression artérielle, mais aussi pour sa constante bonne humeur et ses vaines tentatives de me convertir à gnuplot. Merci Camille pour toutes nos discussions matinales, pour les litres de thé qu'elle *accidentellement* renversé sur la moquette de mon bureau, mais surtout pour sa gentillesse et sa bienveillance hors du commun.

Je tiens ensuite à remercier plusieurs groupes de personnes qui m'ont aidé à préserver ma santé mentale même dans les moments les plus difficiles de cette thèse. Je commence par toutes les personnes avec lesquelles j'ai eu le plaisir de grimper pendant ces trois ans, que ce soit en salle ou à l'extérieur. Deux mentions spéciales vont à la Pangoulou climbing team, Arnaud, Baptiste et Pierre, et à mes premiers compagnons d'escalade ici en région Parisienne, Johann et Coralie. Les entraînements avec vous, et les bières qui parfois (pour ne pas dire souvent) suivaient, pouvaient sauver n'importe quelle journée, aussi mauvaise qu'elle soit. Je remercie aussi mes compagnons

vidéoludiques, auto-proclamés 'Team Yolo Xib', qui m'ont permis de me défouler et de m'amuser pendant des longues soirées/nuitées sur un tas de jeux différents, mais aussi pour leurs discussions interminables sur des sujets aussi divers que bizarres.

Je tiens aussi à remercier tous mes amis habitant trop loin pour faire partie de mon quotidien pendant cette thèse, mais avec lesquels on a tout de même réussi à garder contact. Des mentions spéciales vont aux pauvres malchanceux qui se sont organisés pour venir assister à ma soutenance, mais qui ont été interdits d'accès à l'ONERA à cause des mesures de confinement contre l'épidémie de covid-19: Alessandro, Caterina, Michele e Sara (auto-proclamés les Bagiani), ainsi qu'à Luca, Monica e Pietro. Je suis sûr qu'on trouvera plein d'autres occasions pour fêter ça dignement.

Finalement, les derniers remerciements vont à ma famille, à mon père, à ma mère et à ma sœur. Pour m'avoir soutenu tout au long des mes études, pour avoir toujours été présents, même à distance, pour m'avoir toujours bien conseillé, pour avoir essayé de résoudre tout un tas de problèmes, sans même que je le demande, pour m'avoir toujours motivé à donner le mieux de moi-même, pour votre exemple, pour tout, merci.

« La notion de passoire est indépendante de la notion de trou »
Devise Shadok

« Sometimes Science is more Art than Science, Morty »
R. Sanchez

« If you stand for nothing, Burr, what do you fall for? »
A. Hamilton, according to L.-M. Miranda

« GG easy tutorial »
PACHECO OBLENCIO

« For I am the Concorde of PhD students, loud and unreliable »
J. Pelamatti

« Ladies and gentlemen, this is mambo number 5: »
Lou Bega

Contents

Remerciements / Acknowledgements v

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Surrogate model-based optimization . . . 3
- 1.3 Thesis plan . . . 3

2 Problem statement 5
- 2.1 Introduction . . . 5
- 2.2 Problem definition . . . 6
  - 2.2.1 Variable-size design space optimization problem . . . 6
  - 2.2.2 Optimality conditions . . . 9
  - 2.2.3 Fixed-size mixed-variable design space problem . . . 10
  - 2.2.4 Computational cost of the design problem functions . . . 11
- 2.3 Review of existing methods dealing with mixed-variable and variable-size design space problems . . . 12
  - 2.3.1 Approach 1: Global exploration algorithms . . . 12
    - 2.3.1.1 Algorithms solving fixed-size mixed-variable problems . . . 12
    - 2.3.1.2 Algorithms solving variable-size design space problems . . . 14
  - 2.3.2 Approach 2: Nested optimization algorithms . . . 14
  - 2.3.3 Approach 3: Sequential optimization algorithms . . . 15
  - 2.3.4 Approach 4: Complete exploration of the non-relaxable search space . . . 16
- 2.4 Literature review synthesis . . . 17
- 2.5 Conclusions . . . 18

3 Mixed-variable Gaussian Processes 19
- 3.1 Introduction . . . 19
- 3.2 Gaussian Process surrogate models . . . 24
- 3.3 Mixed-variable Gaussian Process modeling . . . 26
- 3.4 Gaussian Process kernels . . . 28
  - 3.4.1 Kernel operators . . . 30
- 3.5 Discrete kernels . . . 32
  - 3.5.1 Compound Symmetry . . . 32
  - 3.5.2 Hypersphere decomposition kernel . . . 34
  - 3.5.3 Latent variable kernel . . . 35
  - 3.5.4 Coregionalization . . . 36
  - 3.5.5 Discrete kernel comparison . . . 38
- 3.6 Considerations on mixed-variable Gaussian Processes . . . 38

3.6.1 Category-wise and level-wise mixed-variable kernels . . . 38
3.6.2 Surrogate model scedasticity . . . 39
3.6.3 Noisy data modeling . . . 42
3.6.4 Hyperparameter optimization . . . 42
3.6.5 Mixed-variable design of experiments . . . 44
3.7 Modeling performance comparison . . . 44
3.7.1 Benchmark analysis . . . 45
3.7.2 Branin function . . . 47
3.7.3 Augmented Branin function . . . 50
3.7.4 Goldstein function . . . 52
3.7.5 Analytical benchmark N.4 . . . 54
3.7.6 Analytical benchmark N.5 . . . 56
3.7.7 Propulsion performance simulation . . . 57
3.7.8 Thrust frame structural analysis . . . 61
3.8 Error model . . . 63
3.9 Result synthesis . . . 64
3.10 Conclusions . . . 67

4 Mixed-variable Bayesian design optimization 69
4.1 Introduction . . . 69
4.2 Mixed-variable Bayesian Optimization . . . 71
4.2.1 Mixed variable acquisition function . . . 72
4.2.1.1 Objective function oriented infill criterion . . . 72
4.2.1.2 Feasibility oriented infill criteria . . . 74
4.2.1.3 Infill criterion optimization . . . 76
4.3 Applications and Results . . . 76
4.3.1 Benchmark analysis . . . 77
4.3.2 Branin function . . . 79
4.3.3 Augmented Branin function . . . 81
4.3.4 Goldstein function . . . 84
4.3.5 Launch vehicle propulsion performance optimization . . . 85
4.3.6 Launch vehicle thrust frame design optimization . . . 89
4.3.7 Result synthesis . . . 94
4.4 Conclusions . . . 95

5 Bayesian optimization of variable-size design space problems 97
5.1 Introduction . . . 97
5.2 Budget allocation strategy . . . 100
5.2.1 Discarding of non-optimal sub-problems . . . 101
5.2.2 Computational budget allocation . . . 103
5.2.3 Bayesian optimization of remaining sub-problems . . . 104
5.2.4 Algorithm overview . . . 105
5.3 Variable-size design space kernel . . . 106
5.3.1 Sub-problem-wise decomposition kernel . . . 106
5.3.2 Dimensional variable-wise decomposition . . . 107
5.3.3 Variable-size design space Gaussian Process training . . . 109
5.3.4 Infill criterion optimization . . . 109
5.4 Applications and Results . . . 110
5.4.1 Benchmark analysis . . . 110
5.4.2 Variable-size design space Goldstein function . . . 112
5.4.3 Multi-stage launch vehicle architecture optimization . . . 117

5.4.3.1 Liquid propulsion . . . 118
5.4.3.2 Solid propulsion . . . 118
5.4.3.3 Variable-size design space problem formulation . . . 119
5.4.3.4 Optimization results . . . 121
5.4.4 Result synthesis . . . 125
5.5 Conclusions . . . 127

6 Conclusions and perspectives **129**
6.1 Conclusions . . . 129
6.2 Perspectives . . . 131

Bibliography **133**

A Publications and communications **143**

B Proof of validity of the mixed-variable noise handling kernel **145**

C Variable-size design space Goldstein function **147**

List of Figures

2.1 Example of MultiDisciplinary Analysis for launch vehicles 11

3.1 Example of a Gaussian Process prior modeling 25
3.2 Example of a Gaussian Process posterior modeling 26
3.3 Mixed-variable modeling example function 27
3.4 Comparison between independent category-wise modeling and mixed-variable modeling 27
3.5 Example of latent variable mapping for a generic discrete variable characterizing the 'material choice' characteristic. 35
3.6 Category-wise independent modeling of a heteroscedastic mixed variable function . 40
3.7 Estimated optimal variance associated to the two categories of a heteroscedastic example for various data set sizes over 20 repetitions per size 41
3.8 Hypothetical initialization (left) and optimal solution (right) for latent variables hyperparameters characterizing the covariance between material choices. 43
3.9 The 4 discrete categories of the Branin function. 48
3.10 Comparison of various discrete kernels modeling performance on the mixed-variable Branin function for various training data set sizes over 20 repetitions. 49
3.11 Comparison of various discrete kernels modeling performance on the augmented mixed-variable Branin function for various training data set sizes over 20 repetitions. 51
3.12 Comparison of various discrete kernels modeling performance on the mixed-variable Goldstein function for various training data set sizes over 20 repetitions. 53
3.13 Categories of the 4th modeling analytical benchmark. 54
3.14 Comparison of various discrete kernels modeling performance on the fourth analytical benchmark function for various training data set sizes over 20 repetitions. . 55
3.15 Comparison of various discrete kernels modeling performance on the fifth analytical benchmark function for various training data set sizes over 20 repetitions. 56
3.16 Comparison of various discrete kernels modeling performance on the fifth analytical benchmark function for various training data set sizes over 20 repetitions. 57
3.17 Prometheus engine, courtesy of ArianeGroup. 58
3.18 Specific impulse of a launcher engine as a function of the nozzle ratio ϵ and the reductant to oxidant ratio O_F for various combinations of oxidant and reductant. . 59
3.19 Comparison of various discrete kernels modeling performance on the launcher engine specific impulse test-case for various training data set sizes over 20 repetitions. 60
3.20 Ariane 6 PPH launcher aft bay. 61
3.21 Examples of structural responses on the entire thrust frame structure. On the left figure the maximum Von Mises stress is illustrated, while on the right figure the upper interface longitudinal over-flux is shown. 62

3.22 Comparison of various discrete kernels modeling performance on the thrust frame structural analysis test-case. The modeled values are the maximum inner skin Von Mises stress (top) and the upper interface longitudinal over-flux (bottom) over 10 repetitions. 63
3.23 Comparison of various discrete kernels error model on the mixed-variable Branin function for various training data set sizes over 20 repetitions. 65
3.24 Comparison of various discrete kernels modeling MNLL on the thrust frame structural analysis test-case. The modeled values are the maximum inner skin Von Mises stress (top) and the upper interface longitudinal over-flux (bottom) over 10 repetitions. 66

4.1 Schematic representation of the working principle of Bayesian Optimization. $\mathbf{X}$ and $\mathbf{Z}$ contain the continuous and discrete variable data sets, while $\mathbf{y}$ and $\mathbf{g}_1, \dots, \mathbf{g}_{n_g}$ contain the associated objective function and constraint responses. 72
4.2 Example of Bayesian optimization of an unconstrained problem performed with the EI acquisition function over 6 iterations. 73
4.3 Comparison of the convergence rate of various discrete kernels during the BO of the mixed-variable Branin function over 20 repetitions. 80
4.4 Comparison of the convergence rates of the proposed BO algorithms and a penalized mixed-variable GA for the mixed-variable Branin function over 20 repetitions. 80
4.5 Location of the infilled data samples during one repetition of the Branin test-case optimization for various kernel parameterizations. 81
4.6 Comparison of the convergence rate of various discrete kernels during the BO of the augmented mixed-variable Branin function over 20 repetitions. 83
4.7 Comparison of the convergence values of various discrete kernels during the BO of the augmented mixed-variable Branin function over 20 repetitions. 83
4.8 Comparison of the convergence rate of various discrete kernels during the BO of the mixed-variable Goldstein function over 20 repetitions. 85
4.9 Comparison of the convergence value of various discrete kernels during the BO of the mixed-variable Goldstein function over 20 repetitions. 86
4.10 Comparison of the convergence value of various discrete kernels during the BO of the mixed-variable Goldstein function over 20 repetitions. Focus on the mixed-variable kernels. 86
4.11 Multidisciplinary design analysis for a solid rocket engine. 87
4.12 Under or over gas expansion at the nozzle exit 87
4.13 Comparison of the convergence rate of various discrete kernels during the BO of the launch vehicle propulsion performance optimization over 10 repetitions. 89
4.14 Comparison of the convergence value of various discrete kernels during the BO of the launch vehicle propulsion performance optimization over 10 repetitions. 89
4.15 Two examples of possible bottom skirt configurations. On the left figure 144 stringers and 4 frames are included while on the right one only 36 stringers and 2 frames are present. 90
4.16 Comparison of the convergence rate of various discrete kernels during the BO of a launch vehicle thrust frame design over 10 repetitions. 92
4.17 Comparison of the convergence value of various discrete kernels during the BO of launch vehicle thrust frame design over 10 repetitions. 92
4.18 Skin thicknesses characterizing the optimal solution obtained for the design optimization of the launch vehicle thrust frame. 93
4.19 Von Mises stress (left) and overflux (right) characterizing the optimal solution obtained for the design optimization of the launch vehicle thrust frame. 93

5.1 Example of Best-case, Worst-case and Nominal-case functions. 102
5.2 Example of a function optimum range, defined as the interval between the BC and the WC 103
5.3 Example of comparison between optimum ranges between two Sub-Problems (SP) over different iterations. In the left figure, the optimum ranges of the two SP overlap and it is therefore not possible to accurately predict whether one of the two contains the global optimum. In the right figure, instead, the two optimum ranges do not overlap and it is therefore possible to discard the SP2 as it does most likely not contain the global optimum. 104
5.4 Budget allocation strategy for the optimization of variable-size design space problems. 105
5.5 Value range of the feasible objective function for each sub-problem of the variable-size design space Goldstein function. 113
5.6 Comparison of the convergence rate of various VSDSP optimization algorithms during the BO of the mixed-variable Goldstein function over 10 repetitions. The continuous lines represent the SOMVSP alternatives, the dashed lines represent the variable-size design space kernel BO alternatives and the dotted lines represent the independent optimization of each sub-problem. 114
5.7 Comparison of the convergence value of different VSDSP optimization algorithms on the variable-size design space Goldstein function over 10 repetitions. 114
5.8 Comparison of the convergence value of different VSDSP optimization algorithms on the variable-size design space Goldstein function over 10 repetitions. Focus on the variable-size design space kernel based BO algorithms. 115
5.9 Comparison of the number of remaining sub-problems along the optimization process for different parameterizations of the SOMVSP on the variable-size design space Goldstein function over 10 repetitions. 116
5.10 Comparison of the convergence rate of different VSDSP optimization algorithms during the BO of the mixed-variable Goldstein function over 10 repetitions. Focus on the the variable-size design space kernel BO alternatives. 116
5.11 Comparison of the convergence value of different VSDSP optimization algorithms on the variable-size design space Goldstein function over 10 repetitions. Focus on the the variable-size design space kernel BO alternatives. 117
5.12 Schematic MDO representation of the liquid propulsion stage. 119
5.13 Schematic MDO representation of the solid propulsion stage. 120
5.14 Considered launch vehicle architectures (S: Solid propulsion, L: Liquid propulsion). 120
5.15 Value range of the feasible objective function for each sub-problem of the multi-stage launch vehicle architecture design problem. 121
5.16 Comparison of the convergence rate of various discrete kernels during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. The bottom part of the plots represents the number of repetitions which have found a feasible solution at a given iteration. 122
5.17 Comparison of the convergence value of different VSDSP optimization algorithms for the BO of the multi-stage launch vehicle architecture design over 10 repetitions. 123
5.18 Comparison of the convergence rate of different VSDSP optimization algorithms during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. Focus on the SOMVSP methods. The bottom part of the plots represents the number of repetitions which have found a feasible solution at a given iteration. 123
5.19 Comparison of the convergence rate of different VSDSP optimization algorithms during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. Focus on the variable-size design space kernel methods. The bottom part of the plots represents the number of repetitions which have found a feasible solution at a given iteration. 124

5.20 Comparison of the remaining number of sub-problems along the optimization process for different SOMVSP algorithms during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. 124
5.21 Sub-problems towards which the compared VSDSP algorithms convergence over the 10 repetitions of the multi-stage launch vehicle architecture design optimization. 126

List of Tables

3.1 Advantages and weaknesses of discrete kernel parameterizations. The acronyms refer to the following kernels: CS: Compound Symmetry, HS: Hypersphere Decomposition, LV: Latent Variable, CN: Coregionalization. 38
3.2 Characterization of the Goldstein function discrete categories 52
3.3 Variables characterizing the combustion performance simulation test-case 57
3.4 Variables characterizing the thrust frame structural analysis test-case 62

4.1 Characterization of the Goldstein function discrete categories 84
4.2 Variables characterizing the solid rocket engine test-case 87
4.3 Variables characterizing the bottom skirt structural analysis test-case 90

5.1 Defining characteristics of the sub-problems comprising the variable-size design space Goldstein function optimization problem. 113
5.2 Variables characterizing each liquid propulsion stage 118
5.3 Variables characterizing each solid propulsion stage 119
5.4 Defining characteristics of the sub-problems comprising the multi-stage launch vehicle architecture optimization problem. (S: Solid propulsion, L: Liquid propulsion) 121

C.1 Characterization of the variable-dimension search space Goldstein function subproblem N°1 discrete categories 148
C.2 Characterization of the variable-dimension search space Goldstein function subproblem N°2 discrete categories 148
C.3 Characterization of the variable-dimension search space Goldstein function subproblem N°2 discrete categories 149
C.4 Characterization of the variable-dimension search space Goldstein function subproblem N°5 discrete categories 149
C.5 Characterization of the variable-dimension search space Goldstein function subproblem N°6 discrete categories 150
C.6 Characterization of the variable-dimension search space Goldstein function subproblem N°7 discrete categories 150
C.7 Characterization of the variable-dimension search space Goldstein function constraint 151
C.8 Characterization of the variable-dimension search space Goldstein function constraint 151
C.9 Characterization of the variable-dimension search space Goldstein function constraint 151
C.10 Characterization of the variable-dimension search space Goldstein function constraint 151

List of Abbreviations

ACO	**A**nt **C**olony **O**ptimization
ANFIS	**A**daptive-**N**etwork-based **F**uzzy **I**nference **S**ystem
ANN	**A**rtificial **N**eural **N**etwork
B&B	**B**ranch & **B**ound
BO	**B**ayesian **O**ptimization
CEA	**C**hemical **E**quilibrium with **A**pplications
CFD	**C**omputational **F**luid **D**ynamics
CMAES	**C**ovariance **M**atrix **A**daption **E**volution **S**trategies
CN	**C**oregio**n**alization
CS	**C**ompound **S**ymmetry
CW	**C**ategory-**W**ise
DE	**D**ifferential **E**volution
DoE	**D**esign of **E**xperiments
DVW	**D**imensional **V**ariable **W**ise
EGO	**E**fficient **G**lobal **O**ptimization
EI	**E**expected **I**mprovement
ES	**E**volution **S**trategies
EV	**E**xpected **V**iolation
FEM	**F**inite **E**lement **M**ethod
GA	**G**enetic **A**lgorithm
GP	**G**aussian **P**rocess
GLM	**G**eneralized **L**inear **M**odel
GLOW	**G**ross **L**ift-**O**ff **W**eight
HS	**H**yper**S**phere
ICM	**I**ntrinsic **C**oregionalization **M**odel
LEO	**L**ow **E**arth **O**rbit
LHS	**L**atin **H**ypercube **S**ampling
LMC	**L**inear **M**odel of **C**oregionalization
LV	**L**atent **V**ariable
MDA	**M**ulti-**D**isciplinary **A**nalysis
MLP	**M**ulti-**L**ayer **P**erceptron
MNLL	**M**ean **N**egative test **L**og-**L**ikelihood
PLS	**P**artial **L**east **S**quare
PoF	**P**robability of **F**easibility
PSO	**P**article **S**warm **O**ptimization
RBF	**R**adial **B**asis **F**unction
RLV	**R**eusable **L**aunch **V**ehicle
RMSE	**R**oot of the **M**ean **S**quared **E**rror

SMBDO	**S**urrogate **M**odel-**B**ased **D**esign **O**ptimization
SOM	**S**elf-**O**rganizing **M**ap
SOMVSP	**S**trategy for the **O**ptimization of **M**ixed **V**ariable-**S**ize design space **P**roblems
SPW	**S**ub-**P**roblem **W**ise
SVM	**S**upport **V**ector **M**achine
VSDSP	**V**ariable-**S**ize **D**esign **S**pace **P**roblem

List of Symbols

$\langle \cdot \rangle$	Hilbert space inner product
$\mathbf{B}$	Coregionalization matrix
$\mathcal{D}$	Data set
$E[\cdot]$	Expected value
$f(\cdot)$	Objective function
F_y	Objective function domain
F_w	Dimensional variable domain
F_x	Continuous variable domain
F_z	Discrete variable domain
$g(\cdot)$	Constraint function
$\hat{g}$	Constraint function prediction mean
$G(\cdot)$	Constraint function Gaussian process
$GP(\cdot)$	Gaussian process
$\mathcal{H}$	Hilbert space
$k(\cdot)$	Covariance kernel function
$\mathbf{K}$	Covariance matrix
$\mathcal{N}(\cdot)$	Normal distribution
n_g	Number of constraints
$\hat{s}(\cdot)$	Objective function prediction standard deviation
$\hat{s}_g(\cdot)$	Constraint function prediction standard deviation
$\mathbf{w}$	Dimensional variable vector
$\mathbf{W}$	Dimensional variable data set
$\mathbf{x}$	Continuous variable vector
$\mathbf{X}$	Continuous variable data set
$\mathbf{y}$	Objective function values data set
$\hat{y}(\cdot)$	Objective function prediction mean
$Y(\cdot)$	Objective function Gaussian process
$\mathbf{z}$	Discrete variable vector
$\mathbf{Z}$	Discrete variable data set
$\delta(\cdot)$	Kronecker delta function
θ	Surrogate model hyperparameters
μ	Gaussian process mean
σ	Gaussian process variance
$\phi(\cdot)$	Mapping function
$\boldsymbol{\psi}$	Covariance vector

CHAPTER 1

Introduction

1.1 Context

In modern days, complex multidisciplinary systems such as aircraft, automotive vehicles and space launch vehicles are required to provide continuously increasing performances at a reduced cost. In this framework, the early stages of the design process, in which the baseline architecture and design are defined and preliminary sizing is performed, are crucial. Indeed, if a non-optimal baseline architecture is selected, the design optimization iterations might refine the sizing and parameterization of the various sub-systems, but the final design will most likely be sub-optimal. An example of design problem that is considered for illustrative purposes throughout this thesis is the design of launch vehicles. Launch vehicles are very complex systems which have the purpose of injecting payloads (*e.g.,* satellites, scientific equipment, probes) on specific target orbits defined by the mission requirements. These systems are developed in order to work in extreme conditions (*e.g.,* high temperatures, hyper-sonic speed, large accelerations, null external pressure) and they are characterized by the interaction between multiple subsystems (*e.g.,* main engines, boosters, hydraulic systems, turbo-pumps, fairing) as well as multiple disciplines (*e.g.,* propulsion, trajectory, control, structural dynamics). As for most multidisciplinary design problems, the different disciplines characterizing a launch vehicle often present antagonistic effects on the system performance. For instance, from an aerodynamic perspective the optimal solution is characterized by small stage diameters in order to reduce the drag during the atmospheric flight phases. On the other hand, from a structural dynamics standpoint, large diameters are preferable in order to increase the system stability. This results in a complex design process characterized by the necessity of determining optimal multidisciplinary compromises.

As mentioned before, the objective of a complex system design process is to determine the system architecture and associated parameterization which yield the best performance while simultaneously complying with all the mission requirements and constraints. However, the criterion which is used in order to asses the performance of a given system can vary (*e.g.,* total cost, maximum speed, fuel consumption, reliability), and is usually defined *ad hoc* by the user depending on the system application. Within the framework of launch vehicle design, the performance requirements are first of all related to safety constraints, due to the fact that a single system failure can have disastrous consequences, such as the loss of human lives, as well as large economical set-backs. Furthermore, a higher performance for a launch vehicle can also be represented under the form of a lower Gross Lift-Off Weight (GLOW). This is due to the fact that low GLOW values can either yield a considerable reduction of the launch cost, or they can enable the injection of heavier payloads in the target orbit. Alternatively, an increase in performance for a launch vehicle can also enable the access to higher energy orbits, such as interplanetary transfer orbits, which may not have been accessible previously.

Due to the large cost and time requirements of prototype developing and physical testing, a large part of complex system design processes rely on the use of numerical simulations, which allow to predict the performance and feasibility of a given system without requiring to actually build it and test it. However, this kind of design problems usually involve computationally intensive simulations which require large amounts of time and considerable computational resources in order to be performed. Typical examples of computationally intensive simulations which can be encountered within the framework of complex system design are

- Finite Element Models (FEM) [78], which are often required for structural mechanics analyses in order to ensure the integrity of the considered system during its mission.
- Computational Fluid Dynamics (CFD) analyses, which are necessary in order to determine the lift and drag forces acting on the considered system, as well as the induced heat generation.
- Coupled multidisciplinary analyses [12], which are necessary to ensure that the design parameters and state variables associated to different sub-systems and/or disciplines are compatible (*i.e.,* result in a feasible solution) within the scope of multidisciplinary design optimization.

In the presence of computationally intensive functions of the sort, the simulation based optimization of a complex system can be challenging, due to the fact that testing a large number of possible configurations and parameterizations may be unfeasible in terms of computational time and/or resources. This issue becomes even more problematic when discrete choices characterizing, for instance, technological or architectural alternatives are taken into account, due to the fact that a large combination of discrete design choices must be dealt with. Typical examples of such discrete choices which can be encountered within the complex system design framework are the type of material (*e.g.,* steel, aluminum, composite), the type of propulsion (*e.g.,* liquid, solid, hybrid), the wing configuration (*e.g.,* straight, delta, swept) and the inclusion of auxiliary sub-systems and/or technologies (*e.g.,* inclusion of a spoiler in the design of a car). The standard approaches when dealing with discrete choices within a complex system design framework consist in either relying on previous designs and expertise in order to downselect the system architecture, or in selecting a few promising alternatives and separately optimizing each one of them. Both alternatives cannot ensure an optimal final design, and often result in the need of design iterations in order to converge towards a final design which complies with all the requirements. Alternatively, the discrete choices can be treated as design variables characterized by a discrete nature (*i.e.,* finite number of possible values, possibly unordered) and included within the system design optimization process. By formulating the design problem in this fashion, various algorithms algorithms allowing to optimize the resulting mixed-variable (mixed continuous/discrete) problems, such as evolutionary heuristic algorithms [35], [73], [74], grid-search based algorithms [4], [9], [79] and surrogate-model assisted algorithms [16], [73], can be considered. However, most of the existing algorithms may be inadequate when dealing with computationally intensive problems, due to the large number of function evaluations required in order to converge to the global feasible optimum. Furthermore, when including discrete technological and architectural choices within the design optimization process, an additional challenge may arise, represented by the fact that depending on the value of some specific discrete variables, the number and type of design variables a given candidate solution depends on as well as the number and type of constraints it is subject to may vary. For instance, when considering the design of a launch vehicle, depending on whether solid or liquid propulsion is selected, the design variables associated to the engine vary, and similarly the constraints to be considered are different as well. Part of the design variables, such as the geometric parameters of the solid propellant grain, are therefore only present if some specific discrete choices are made. The resulting problems are sometimes referred to as variable-size design space problems [93]. When dealing with this kind of problems, only a few heuristic [2], [3], [93] and

grid-search based algorithms [4], [9], [79] exist in the literature. Also in this case, however, the existing algorithms tend to be inadequate in the presence of computationally intensive problems due to their low convergence speed, which is often not compatible with the limited simulation budget.

1.2 Surrogate model-based optimization

Among the existing families of optimization methods, a promising candidate which can potentially allow to perform the global optimization of mixed-variable and variable-size design space problems while also requiring a limited amount of function evaluations are the Surrogate Model-Based Design Optimization (SMBDO) algorithms [105]. These algorithms rely on using surrogate models (*i.e.,* analytical approximations) of the considered problem objective and constraint functions, which are generally characterized by a negligible computational cost, in order to iteratively determine and explore the most promising locations of the design space, thus simultaneously refining the surrogate model and converging towards the problem optimum. A number of different SMBDO algorithms allowing the optimize purely continuous optimization problems (*i.e.,* with objective and constraint functions depending solely on continuous design variables) exist in the literature. Depending on the considered algorithm, different surrogate modeling techniques can be considered. Popular examples are the Radial Basis Function (RBF) [56], the Support Vector Machine (SVM) [118] and the Moving Least Square ([68]). Furthermore, depending on the considered algorithm the design space exploration criterion can also vary.

Although the SMBDO of continuous problems is a popular research domain, only a few adaptations of this kind of methods for the optimization of problems characterized by the presence of discrete design variables exist [16], [37], [60]. Furthermore, most of the proposed methods do not offer a solution for the mixed-variable problem in its most generic formulation. Finally, the constraint handling of these SMBDO techniques is usually penalization-based and can therefore be inefficient (in terms of necessary function evaluations) in case the weights are not properly tuned. For these reasons, the main objective of this thesis consists in adapting and extending surrogate model-based design optimization algorithms in order to allow the optimization of constrained mixed-variable and variable-size design space problems with a limited number of function evaluations, thus providing a potentially useful tool for the integration of discrete technological and architectural choices within the computationally intensive design of complex systems. More specifically, the SMBDO methods which are considered and developed in this thesis are based on the Bayesian optimization algorithm [63], characterized by the use of Gaussian process surrogate modeling [107]. Throughout this work, optimization problems and test-cases related to the design of launch vehicles are considered in order to better highlight the engineering related applications (and associated challenges) of the discussed optimization methods. However, it is important to note that the topics of this thesis are actually applicable to a much wider range of design problems.

1.3 Thesis plan

Including the present introduction, the manuscript is divided into 6 main chapters:

In **Chapter** 2, the inclusion of discrete choices within the design process is generalized and properly formulated under the form of a mixed-variable variable-size design space problem. Subsequently, a review of the existing approaches and algorithms allowing to solve this kind of problem, either in its global or simplified form are discussed, and their limitations are highlighted.

In **Chapter** 3, the Gaussian process based surrogate modeling of mixed continuous/discrete

functions is discussed. More specifically, the definition of mixed-variable kernels as a combination of purely discrete and purely continuous kernels is described. Subsequently, a unified formalism allowing to define and validate the existing discrete kernels is proposed, thus facilitating the comparison between the various parameterizations from a theoretical perspective. Finally, the discrete kernel are extensively compared on a number of analytical and engineering related test-cases with various parameterizations, thus allowing to better highlight their strengths and limitations.

In **Chapter** 4, the possibility of extending the concept of Bayesian optimization in order to solve constrained problems characterized by continuous and discrete variables (but a fixed-size design space) by relying on the mixed-variable Gaussian process modeling presented in Chapter 3 is discussed. More speficially, it is shown that purely continuous acquisition functions (for both objective and constraints functions) can be applied to the mixed-variable case under the condition that the Gaussian process kernels are properly constructed, and the challenges associated to their optimization within the mixed-variable design space are described. Finally, the proposed mixed-variable Bayesian optimization algorithm is tested with different discrete kernel parameterizations on a number of analytical and engineering related test-cases.

In **Chapter** 5, two alternative extensions of the mixed-variable Bayesian optimization algorithm proposed in Chapter 4 allowing to solve variable-size design space optimization problems are discussed and tested on analytical and engineering related test-cases. The first alternative is based on the separate optimization of various mixed-variable fixed-size sub-problems coupled with a computational budget allocation strategy relying on the information provided by the various sub-problems surrogate models. The second alternative, instead, is based on the definition of a Gaussian process kernel allowing to compute the covariance between data samples defined in partially different search spaces by relying on a hierarchical grouping of design variables.

In **Chapter** 6, a global synthesis of the analyses, methods and algorithms presented in this thesis is provided and subsequently the most relevant conclusions are drawn. Finally, possible improvements, extensions and perspectives of the presented work are discussed.

The research work presented in this thesis is based on the published work and on the communications which are listed in Appendix A.

CHAPTER 2

Problem statement

2.1 Introduction

The design of complex systems, such as launch vehicles, aircraft, automotive vehicles or electronic components, can usually be represented under the form of an optimization problem. In other words, for a given formulation of the problem in terms of objective and constraint functions, as well as design variables these functions depend on, the aim of the design process is to determine the values of the design variables which yield the best value of the objective function. Furthermore, this optimal solution must also comply with all the constraints the problem is subject to. In the most simple case, the architecture of the considered system is determined beforehand through an empirical process. As a consequence, the design variables characterizing the resulting problem are usually defined within a continuous search space. For instance, within the framework of Reusable Launch Vehicle (RLV) design, typical continuous design variables are sizing parameters, combustion pressures and propellant masses. However, in most real world engineering design problems, the architecture of the considered system cannot be specified beforehand and must typically be defined during the early stages of the design process. In this case, it is therefore necessary to simultaneously optimize the architecture layout and the continuous design variables which characterize it. The most straightforward approach in order to perform this simultaneous optimization consists in iterating over every possible architecture definition and performing an optimization of the continuous design variables for each architecture [102]. However, depending on the complexity of the considered system, it may be necessary to consider several thousands of different architectures [45], thus resulting extremely time consuming, if not unfeasible.

Without loss of generality, the choices related to the architecture definition can be represented under the form of discrete design variables, sometimes referred to as categorical or qualitative variables, which characterize design alternatives and/or technological choices. Typical examples of technological choices which can be encountered within the context of RLV design are the type of material to use for a given sub-system, the type of propulsion to include in a given stage, the presence of a certain system component and the number of reinforcements to be included in a given structure. From an analytical perspective, part of these technological choices plays the role of standard discrete variables defined within a finite set of choices, whereas others play a more complex role, as they can also influence the definition of the objective and constraint functions, as well as the number and type of variables that characterize the problem. These particular choices are often related to the selection of sub-problem specific technologies. For instance, depending on whether a combustion, hybrid or electric engine is selected for the design of a car, the specific engine related design variables can vary considerably (*e.g.,* combustion chamber pressure, electric engine torque). Additionally, the associated constraint can also differ (*e.g.,* combustion temperature associated constraints). Due to the inherent discrete and potentially non-numerical nature

of these design variables, the concept of metrics is usually not definable within their domain, thus resulting in an unordered set of possible choices. In the literature, this kind of problems are referred to as **mixed-variable optimization problems** [79] or **Variable-Size Design Space Problem** [93] (VSDSP). Most modern optimization algorithms are developed with the purpose of solving design problems essentially characterized by continuous and integer variables and by consequence, the introduction of these categorical (qualitative) variables raises a number of additional challenges. Firstly, a large number of algorithms cannot be used due to the non-differentiability of the objective function and the possible absence of metrics in the discrete variables domains (*e.g.,* gradient-based algorithms and Nelder-Mead simplex [92]). Further limitations arise from the need to initialize newly created variables during the optimization process in a dynamically varying design space (*e.g.,* Genetic algorithms [48]) and the impossibility to relax the integrity constraint on some of the discrete variables (*e.g.,* Branch and Bound (B&B) [69]). Finally, for problems characterized by large numbers of technological choices and options, the combinatorial size of the discrete variables search space can be considerably large, thus rendering a complete exploration of the problem search space computationally particularly difficult. Moreover, when this issue is coupled with computationally intensive problem functions (in terms of computation time) a complete exploration of the problem search space might become unfeasible. The objective of this chapter is to present the analytical definition of the variable-size design space problem and provide an analysis of the additional challenges that can arise when solving them. Furthermore, a brief review of the existing optimization algorithms allowing to solve variable-size design space problems is provided. In the first section of this chapter, the analytical definition of the variable-size design space problem is provided, and is followed by the fixed-sized derivation. Subsequently, the main challenges related to the optimization of such problems are discussed. In the second part of the chapter, instead, the existing algorithms allowing to deal with variable-size design space problems are described and categorized as a function of the approach they rely on in order to handle discrete variables. Finally, the advantages, the weaknesses and the limitations of the presented algorithms are discussed, which allows to highlight the need for the methods developed in this thesis.

2.2 Problem definition

2.2.1 Variable-size design space optimization problem

Without loss of generality, a generic variable-size design space optimization problem can be modeled as depending on three different types of design variables: continuous, discrete and dimensional.

- **Continuous variables**: **x**
  Continuous variables refer to real numbers defined within a given interval. Typical examples of continuous variables which can be encountered within the framework of complex system design are structure sizing parameters, combustion pressures, propellant masses and time related design parameters.

- **Discrete variables**: **z**
  Discrete variables are non-relaxable variables defined within a finite set of choices. Typical examples of discrete variables which can be encountered within the framework of complex system design are the choice of material, the choice of propulsion, architectural and technology alternatives, number of structural reinforcements and number of engines. Discrete variables are typically divided into 2 categories: quantitative and qualitative. As the name suggests, quantitative variables (sometimes also referred to as ordinal) are related to measurable values and by consequence, a relation of order between the possible values of a given variable can be defined (*i.e.,* it is possible to determine whether a value is larger, smaller

or equal to another). Quantitative discrete variables are often associated to integer variables, although it is not a necessary requirement. When dealing with qualitative variables, instead, no relation of order can be defined between the possible values of a given variable. For instance, if the considered variable characterizes the type of material to be used for a given system structure, it is not possible to determine whether a possible choice (*e.g.,* steel, aluminum, titanium, composite material) is larger, smaller or equal to another. Qualitative variables are sometimes also referred to as categorical or nominal variables. For clarity and synthesis purposes, no distinction between quantitative and qualitative variables is made in this thesis, and both types of variables are simply referred to as **discrete variables**. Finally, it is worth mentioning that some of the discrete variables considered in this work may technically handled by relaxing the integrality constraint, as is for instance proposed in [15] in combination with a B&B optimization algorithm. However, in the remainder of this work it is assumed that none of the discrete variables are relaxable in order to provide a generally applicable solution for the optimization of variable-size design space problems.

- **Dimensional variables**: **w**
  Similarly to the discrete variables, dimensional variables are non-relaxable variables defined within a finite set of choices. The main distinction is that, depending on their values, the number and type of continuous and discrete variables the problem functions depend on can vary. Furthermore, they can also influence the number and type of constraints a given candidate solution is subject to. These particular variables often represent the choice between several possible sub-system architectures. Each architecture is usually characterized by a partially different set of design variables, and therefore, depending on the considered choice, different continuous and discrete design variables must be optimized. For illustrative purposes, a few examples of dimensional variables as well as the associated continuous and discrete design variables and constraints which may be encountered within the framework of RLV design are discussed.

  – Propulsion
    Each stage of a launch vehicle can be characterized by different types of propulsion, *i.e.,* solid, liquid and hybrid. Due to the considerably difference in nature between these technologies, the design variables associated to the propulsion sub-system can be very different. Notable examples are the type of reductant and oxydant (*i.e.,* propellant) the engine combustion relies on. Depending on the selected type of propulsion, the set of possible propellant choice differs. Additionally, each type of propulsion is associated to a set of technology specific variables. For instance, if solid propulsion is considered, the shape of the propellant grain (*i.e.,* circular, star-shaped) as well as the specific shape sizing must be optimized.

  – Material
    The structure of the various sub-systems of a launch vehicle can be either made of metallic or composite material. While the first option is mainly associated to a specific choice of material and geometrical sizing parameters, composite materials can be associated to the matrix and fiber choices as well as integer and continuous parameters such as the number and the orientation of composite plies. Furthermore, the number and type of structural integrity constraints associated to each choice of material is also considerably different. For instance, compression related structural stresses are taken into account differently depending on whether metal or composite material is considered.

– Flight configuration
  Among the alternative solutions which can allow to re-use launch vehicles, one of the options consists in including lifting surfaces in the system design (*e.g.,* wings), thus enabling the launch vehicle to glide back to the landing area. If the option of including lifting surfaces in the design is selected, a number of design variables associated to these lifting surfaces, such as shape and sizing parameters, must be optimized. Furthermore, additional constraints necessary to ensure the structural integrity of the lifting surfaces must also be complied with.

– Trajectory
  The trajectory followed by the launch vehicle in order to reach the target orbit is usually optimized together with the launch vehicle itself. Depending on the choices which are made when defining the trajectory, design variables such as the presence and duration of a ballistic phase and the number and duration of burns for the return to the launch site may need to be optimized.

Although both discrete and dimensional variables may lack a conceptual numerical representation (*i.e.,* they are not associated to measurable values), it is common practice to assign an integer value to every considered alternative of the given variable in order to provide a clear problem formulation as well as simplifying the implementation process [37]. For instance, if 3 hypothetical choices for the type of material are considered (*e.g.,* aluminum, steel, titanium), the associated discrete design variable can be defined as having 3 possible values: 0,1 and 2. In this work, it is assumed that none of the discrete and dimensional variables are directly related to a measurable value and the integer values associated to the possible discrete choices are assigned arbitrarily. However, it is important to highlight the fact that this choice has no influence on working principle of the various algorithms presented in the following sections. Similarly to what is proposed by Lucidi *et al.* [79], a generic variable-size design space problem can be formulated as follows:

$$
\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}) \qquad f : \mathbb{R}^{n_x(\mathbf{w})} \times \prod_{d=1}^{n_z(\mathbf{w})} F_{z_d} \times F_w \rightarrow F_f \subseteq \mathbb{R} \\
\text{w.r.t.} \quad & \mathbf{x} \in F_x(\mathbf{w}) \subseteq \mathbb{R}^{n_x(\mathbf{w})} \\
& \mathbf{z} \in \prod_{d=1}^{n_z(\mathbf{w})} F_{z_d} \\
& \mathbf{w} \in F_w \\
\text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0 \\
& g_i : F_{x_i}(\mathbf{w}) \times \prod_{d=1}^{n_{z_i}(\mathbf{w})} F_{z_{d_i}} \times F_w \rightarrow F_{g_i} \subseteq \mathbb{R} \qquad \text{for} \qquad i = 1,...,n_g(\mathbf{w})
\end{aligned}
\tag{2.1}
$$

where $f(\cdot)$ and $\mathbf{g}(\cdot)$ are respectively the objective function and constraint vector, $\mathbf{x}$ is a $n_x(\mathbf{w})$-dimensional vector containing the continuous design variables, $\mathbf{z}$ is a $n_z(\mathbf{w})$-dimensional vector containing the discrete design variables, $\mathbf{w}$ is a n_w-dimensional vector containing the dimensional design variables and $n_g(\mathbf{w})$ is the number of constraints the problem is subject to. For the sake of clarity, the discrete variables domain is referred to as $F_z(\mathbf{w})$ in the remainder of the thesis:

$$
F_z(\mathbf{w}) = \prod_{d=1}^{n_z(\mathbf{w})} F_{z_d} \tag{2.2}
$$

Similarly, the variable domains relative to the constraint g_i are referred to as $F_{z_i}(\mathbf{w})$ and $F_{x_i}(\mathbf{w})$. Finally, it is important to highlight that throughout this thesis, only single-objective optimization

is considered.

As can be noticed, the continuous and discrete search spaces of Eq. 2.1 are not fixed throughout the optimization process, as they depend on the values of the dimensional variables. As a result, two different candidate solutions of the optimization problem defined above can be characterized by a partially different set of variables, depending on the values of their dimensional variables. In the same way, the feasibility domain can also vary according to the values of $\mathbf{w}$, which results in different candidate solutions possibly being subject to different constraints (in terms of both type and number of active constraints). Furthermore, it is important to notice that the constraint functions can also present a search space which can vary as a function of the dimensional variable values. By consequence, the optimization problem defined in Eq. 2.1 varies **dynamically** along the optimum search, both in terms of number and type of design variables as well as feasibility domain, depending on the values of the candidate solution dimensional variables. For the sake of clarity, the possibility of having the dimensional variables search space vary as a function of the dimensional variables themselves is not taken into consideration. Although this choice does not limit the applicability of the methods presented in this thesis it could, however, require a more complex problem formulation. It is essential to highlight the fact that although the design space may vary depending on the dimensional variable values, the quantity represented by the objective function (*e.g.,* the lift-off weight of a launch vehicle in kg, total production cost of a car in €) remains the same throughout the optimization process.

For the sake of conciseness, in the remainder of this thesis the following two terms are used:

- **levels**: the possible values of a discrete ($z \in \{z_1, \dots, z_l\}$) or a dimensional ($w \in \{w_1, \dots, w_l\}$) variable, where l indicates the total number of levels associated to a given variable
- **categories**: possible combinations of levels. In other words, a category characterizes a candidate solution in the combinatorial discrete/dimensional search space.

As an illustrative example, 3 hypothetical discrete variables associated to the RLV design can be considered:

$$\mathbf{z} = \{z_1, z_2, z_3\} : \begin{cases} z_1 \in \{\text{aluminum}, \text{steel}, \text{titanium}\} \\ z_2 \in \{\text{liquid propulsion}, \text{solid propulsion}\} \\ z_3 \in \{1 \text{ engine}, 2 \text{ engines}, 4 \text{ engines}, 8 \text{ engines}\} \end{cases}$$

In this case, the choice *aluminum* is one of the levels of the variable z_1, and the 3 discrete variables are respectively characterized by 3, 2 and 4 levels. A possible category of the problem above can, for instance, be characterized by {steel, liquid propulsion, 8 engines}. Let l_d be the number of levels associated to the discrete variable z_d, the total number of categories m associated to a given problem can be computed as:

$$m = \prod_{d=1}^{n_z} l_d \tag{2.3}$$

For instance, the number of categories associated to the illustrative example above is equal to $m = 3 * 2 * 4 = 24$.

2.2.2 Optimality conditions

Following the problem formulation provided in Eq. 2.1, it is possible to define the optimal solution, corresponding to the feasible global minimum of the objective function, determined by

a set of points $\{\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}^*\}$ such that:

$$\begin{aligned} & f(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}^*) \leq f(\mathbf{x}, \mathbf{z}, \mathbf{w}) \quad \forall \mathbf{w} \in F_w, \ \forall \mathbf{z} \in F_z(\mathbf{w}), \ \forall \mathbf{x} \in F_x(\mathbf{w}) \\ & \text{s.t.} \quad \mathbf{g}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}^*) \leq 0 \end{aligned} \tag{2.4}$$

In order to determine whether a candidate solution yields the global optimal value of a function, a preliminary step usually requires determining whether the aforementioned point is a local minimum of the objective function. For problems characterized solely by continuous variables, the definition of a local minimum is clearly defined and relies on the definition of a ball B_ε around the point that is being considered:

$$\begin{aligned} & \exists \ \varepsilon > 0 : \forall \ \tilde{\mathbf{x}} \in B_\varepsilon(\mathbf{x}^*) \cap \mathbb{R}^{n_x} \quad f(\tilde{\mathbf{x}}) \geq f(\mathbf{x}^*) \\ \text{with} \quad & B_\varepsilon(\mathbf{x}^*) = \{\tilde{\mathbf{x}} \in \mathbb{R}^{n_x} : \ ||\tilde{\mathbf{x}} - \mathbf{x}^*|| \leq \varepsilon\} \\ \text{s.t.} \quad & \mathbf{g}(\tilde{\mathbf{x}}) \leq 0 \end{aligned} \tag{2.5}$$

For optimization problems involving dimensional and discrete variables, this definition of local minimum is not applicable due to the absence of metrics in the discrete and dimensional search space. It is sometimes possible to define a local minimum with respect to the neighborhood of the candidate solution by relying on problem specific knowledge, as is discussed in [4], [9] and [79]. However, in general no universal definition of neighborhood exists for VSDSP and an *ad-hoc* definition of the optimality conditions is necessary. This is related to the fact that no *a-priori* relation of order exists between the objective function values obtained for different levels of the discrete and dimensional variables and therefore no direct method exists to determine which points must be evaluated in order to assess the local optimality of a candidate solution. It is, however, very challenging to determine an *ad-hoc* definition of neighborhood and, unless problem specific knowledge is included, the only way of ensuring that a given candidate solution represents the global or local optimum of the considered problem is to optimize the continuous variables associated to every category of dimensional/discrete design space. In practice, this is equivalent to considering the non continuous part of the neighborhood of each point as being comprised of the entire discrete/dimensional combinatorial design space. If this approach is used, the neighborhood of a given point is considered to be the entire search space of the problem. However, this thesis lies within the framework of computationally expensive problems, as is discussed in Section 2.2.4, and by consequence, this type of approach becomes either unfeasible or highly inefficient.

2.2.3 Fixed-size mixed-variable design space problem

In the process of proposing and developing methods and approaches allowing to efficiently optimize VSDSP, it might be necessary to first solve a simplified fixed-sized version of this type of problems, characterized by the absence of dimensional variables in their design space. This allows to better understand how to deal with the challenges associated to the presence of discrete variables within the framework of computationally intensive optimization problems, while still dealing with a fixed-sized design space and feasibility domain. Under this assumption, the optimization problem defined in Eq. 2.1 can be re-written in order to obtain a so-called fixed-sized mixed-variable problem:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{z}) \qquad f : F_x \times F_z \rightarrow F_f \subseteq \mathbb{R} \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x} \\ & \mathbf{z} \in F_z \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq 0 \\ & g_i : F_{x_i} \times F_{z_i} \rightarrow F_{g_i} \subseteq \mathbb{R} \qquad \text{for} \qquad i = 1, ..., n_g \end{aligned} \tag{2.6}$$

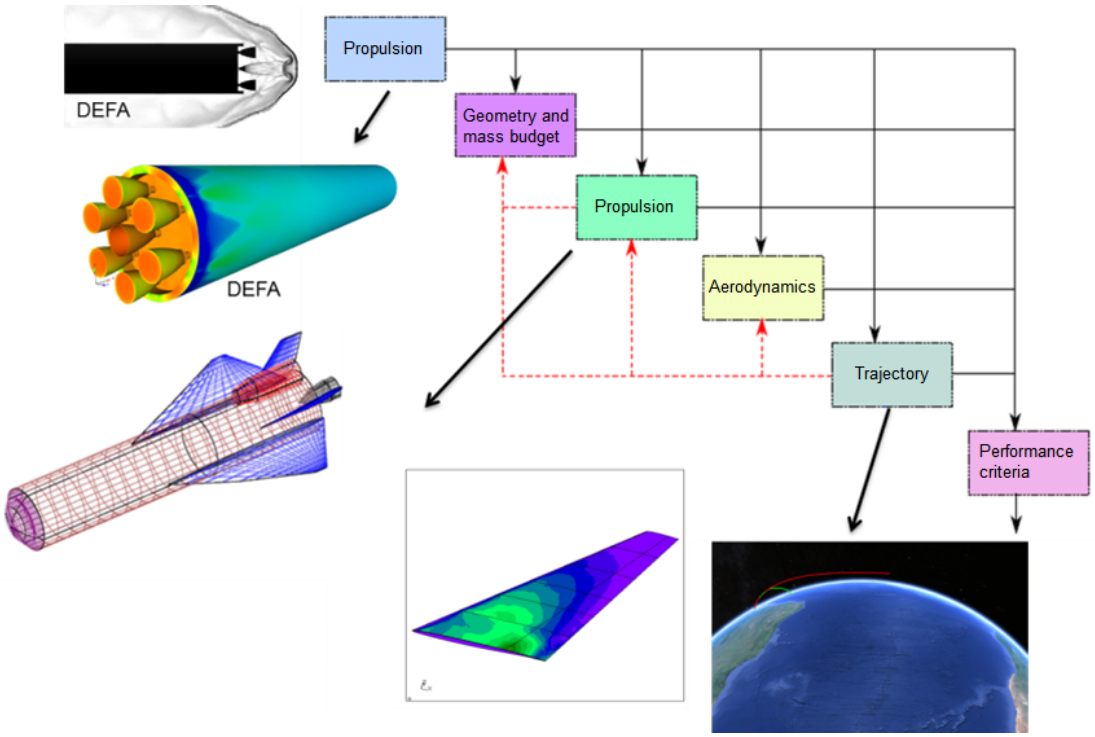


FIGURE 2.1: Example of MultiDisciplinary Analysis for launch vehicles

where the definitions of the remaining elements are identical to the ones provided for Eq. 2.1. It can be noticed that in the optimization problem defined in Eq. 2.6, both the design space and the feasibility space are *fixed* as their definition does not depend on the value of dimensional variables. By consequence, each candidate solution of the optimization problem defined above depends on the same number and type of variables, and is subject to the same constraints.

In the remainder of the thesis, the general optimization problem defined in Eq. 2.1 is referred to as **Variable-Size Design Space Problem**, whereas the fixed-sized simplification is simply referred to as **mixed-variable optimization problem** (or alternatively mixed continuous/discrete optimization problem).

2.2.4 Computational cost of the design problem functions

As is mentioned in Chapter 1, the main focus of this thesis is towards the design optimization of complex engineering systems, such as launch vehicles, aircraft and automotive vehicles. Within this framework, the objective and constraint functions which characterize the considered optimization problem often require a large computational effort to be evaluated. In most cases, this translates into a long computational time, which considerably limits the possibility of exploring the design space at will. Typical examples of computationally intensive functions which can characterize design problems are structural stress calculations based on Finite Element Method (FEM) models [78], Computational Fluid Dynamics (CFD) analyses, fixed point iterations and iterative Multidisciplinary Design Analyses (MDA) [13]. For instance, the design of RLV involves several disciplines and is customarily decomposed into interacting sub-models for propulsion, aerodynamics, trajectory, mass and structure, as is schematically shown in Figure 2.1. The launch vehicle performance estimation, which results from flight performance, reliability and cost, requires coupled disciplinary analyses [14]. The different disciplines are a primary source of trade-offs due to their opposing effects on the launcher performance, and finding a feasible compromise between disciplines can therefore be computationally intensive. The approaches proposed in this manuscript are developed under the assumption that the entirety of the objective

and constraint functions characterizing the considered problems are computationally intensive to evaluate. Under this assumption, the driving objective of the thesis is to develop optimization methods for VSDSP and Mixed-variable problems which provide fast convergence towards global optima (or at least optima neighborhoods) in terms of necessary number of evaluations of the problem functions. For the same reasons, the comparisons between algorithms which are presented in this work are performed by allocating a fixed and identical computational budget (in terms of function evaluations) to all the algorithms. In practice, the methods are compared with respect to the feasible objective function value they yield with a fixed amount of function evaluations rather than by assessing the number of iterations required to reach a given feasible objective function threshold (*e.g.,* optimum neighborhood). Finally, it is assumed that the computational overhead required by some of the proposed methods (*i.e.,* calculations required during the optimization process aside from the evaluation of the problem function values) is negligible when compared to the computationally intensive objective and constraint functions. As is discussed in the following sections of this thesis, this last assumption is particularly important when dealing with the creation and training of surrogate models of the problem functions.

2.3 Review of existing methods dealing with mixed-variable and variable-size design space problems

Several different algorithms allowing to solve VSDSP and/or mixed-variable problems are proposed in the literature. In this section, the different existing approaches for the optimization of such problems are described and discussed, and the relevant related algorithms are briefly presented. Please note that for the sake of clarity and synthesis, the review presented in the following paragraphs does include mixed-variable Surrogate-Model Based Design Optimization methods, as they are discussed in the introduction of Chapter 4. The following review is an extension of the work presented in [97].

2.3.1 Approach 1: Global exploration algorithms

The first and most straightforward existing approach to VSDSP consists in optimizing all the design variables within the same optimization process with a single algorithm. In other words, this approach regroups the algorithms which directly optimize VSDSP as defined in Eq. 2.1. The possibility of relying on this solution depends on the optimization algorithm that is being used as well as on the characteristics of the considered problem. Generally speaking, the algorithms relying on this approach have the advantage of exploring the search space in an efficient and selective way while usually also involving a relatively straightforward implementation. However, most of the algorithms existing for VSDSP have been adapted from either continuous or integer optimization algorithms. By consequence, depending on the specific characteristics of the problem, the adaptations of the optimization algorithms might not perform as well as required for all different types of design variables and might therefore not yield the expected results. For clarity purposes, the discussed algorithms are separated in two categories depending on whether they handle problems with variable-sized search space or not.

2.3.1.1 Algorithms solving fixed-size mixed-variable problems

In case problems with fixed-size mixed continuous/discrete search spaces are considered, as defined in Eq. 2.6, heuristic algorithms such as the Evolution Strategy (ES) [18] can be adapted in a fairly straightforward manner by modifying the type of encoding that is implemented and by taking into account the discrete nature of part of the design variables during mutation, crossover and selection processes. For instance, this can be achieved by including discrete probability distributions in order to model the mutation processes of discrete variables. A first example of

this approach is the adaptation of ES discussed by M. Emmerich *et al.* [35] in the context of the optimization of chemical plants design. In this case, the recombination process is performed according to a domination logic, where every non-relaxable design variable is selected with equal probability from one of the corresponding parents parameters. The mutation of the discrete variables, instead, occurs by randomly choosing the mutated value from the feasible search space. With similar mutation and cross-over adaptations, a mixed-variable Genetic Algorithm [48] can be found in [121]. The possibility of coupling the previously described mixed-variable ES algorithm with a surrogate problem generated through Radial Basis Function Network (RBFN) is discussed by R. Li *et al.* [73]. The main idea presented in this paper is to evaluate each generation of the ES on a surrogate model of the problem functions, thus reducing the overall computational cost of the optimization, and evaluate the actual problem functions only on the most promising individuals of the population at the given generation. In order to deal with qualitative design variables, the RBFN is constructed with respect to a binary encoding of these variables. Particle Swarm Optimization (PSO) [65] and Ant Colony Optimization (ACO) [31] algorithms have also been adapted in order to enable the solution of mixed-variable problems. Although a formulation of these methods for integer variables can be easily defined, a dedicated adaptation for mixed-variable problems is more challenging due to the fact that the considered algorithms heavily rely on the concept of distance between candidate solutions. An adapted ACO formulation for mixed-variable problems is proposed by T. Liao [76]. In this algorithm, the continuous and integer variables are processed in a similar fashion as in the original ACO formulation, while the categorical variables values are determined at each iteration through a random sampling. The probability of a given value to be selected depends on both the number and the performance of the solutions present in the archive that use the given value as well as on the number of values for the same variable that have not yet been tested. In a similar way, an adaptation of PSO is described by C. Liao *et al.* [74] in order to process categorical variables for the optimization of scheduling problems. This is achieved by redefining the particle velocity in the non-relaxable discrete search space as a probability for the particle to explore the feasible values in the position neighborhood. An alternative approach is suggested by C. Sun *et al.* [124], consisting in allowing each particle to travel a distance equal to the smallest possible discrete step between subsequent iterations. In this case, the new value of the particle discrete coordinates is chosen among the feasible neighborhood of the previous position. In [129], instead, PSO is performed by first testing all the levels of the categorical variables through random selection. Subsequently, the discrete variable levels are ranked as a function of both their performance and their compliance with the constraints. Finally, during the final stage of the optimization process the values for the discrete variables are chosen through a random process based on their ranking placement. It is important to note that in this approach, the ranking is performed according to the performance of the single discrete variables levels rather than according to the performance of the discrete category. By consequence, it might perform poorly when confronted with problems characterized by a large combinatorial search space. A variant of the Differential Evolution (DE) algorithm [122] allowing to optimize problems depending on continuous, integer and discrete variables is proposed in [67]. In this case, the integer variables are handled through truncation, whereas the discrete ones are handled by representing them through an index assigned to each level of the various discrete variables. In practice, instead of directly optimizing the discrete variable value, an integer index assigned to the aforementioned discrete variable is optimized. However, it must be pointed out that this approach implies the existence of a relation of order between the discrete variable levels and might therefore perform poorly if it is not the case. In some specific cases, the discrete design variables can be associated to a finite set of continuous values which can be relaxed during the optimization problem. For instance, in [15] a Branch and Bound (B&B) variant allowing to optimize a mixed-variable structure problem is proposed. The underlying concept relies on replacing the choice of shape and material by their associated continuous physical properties, such as tensile strength and Young modulus, and subsequently performing the B&B optimization

by relaxing these properties.

2.3.1.2 Algorithms solving variable-size design space problems

In case VSDSP are considered, a number of additional challenges such as the dynamically varying design space, the initialization of newly created design variables and the presence of a varying number of constraints must be taken into account. Among the various heuristic algorithms which could be adapted in order to solve VSDSP, a promising candidate by its inherent nature is the GA, as different types of variables can be included by simply considering different encodings for every design variable type that characterizes the problem at hand. Furthermore, the standard GA formulation does not make use of either the concept of distance or the derivative of the objective and constraint functions and avoids therefore a number of problems related to the unordered nature of the discrete and dimensional search spaces. In [3], O. Abdelkhalik introduces the *hidden gene* adaptation of GA for the optimization of inter-planetary trajectories with variable number of gravitational assists, resulting in the appearance and disappearance of variables describing the orbital maneuvers. In the proposed algorithm, each candidate solution is represented by a chromosome containing the entirety of genes that can characterize the considered problem. However, not all the genes are taken into account when computing the value of the objective and constraint functions. The choice regarding which genes are considered and which genes are hidden (*i.e.,* with no influence on the problem functions) depends on the values of a limited number of so-called activation genes. This variant of GA has the advantage of being intuitive and easily implementable. However, cross-overs and mutations over hidden parts of the vector result in unproductive numerical operations, thus wasting computational effort. Furthermore, for problems characterized by a large number of discrete categories, the vector containing the entirety of possibly present variables might become considerably large and therefore inefficient memory-wise. In the same way, O. Abdelkhalik also proposed the implementation of the hidden gene approach within DE for a similar inter-planetary trajectory planning problem [2]. A more complex, but theoretically more efficient adaptation of GA called the Structured-Chromosome Evolutionary Algorithm is proposed by Nyew *et al.* in [93]. In this algorithm, the candidate solution is conceptually represented by a hierarchical multi-level chromosome structure rather than a linear one. Differently than in the standard formulation of GA, the genes of each chromosome are linked by both vicinity and hierarchy relationships. For this reason, the encoding of the single variables becomes more complex as it also includes pointers to both the immediate neighbors at the same level and pointers to eventual children genes. Furthermore, in case the mutation of a dimensional variable gene results in the appearance of additional design variables, it is necessary to ensure that the newly created genes are of the correct type and comply with the problem specifics. Compared to the hidden genes approach, this solution has the advantage of not performing computationally wasteful mutations and cross-overs. However, the encoding of the individual chromosomes is much more complex and often requires problem-specific knowledge in order to be efficiently implemented. Finally, it is worth mentioning that all of the optimization algorithms discussed in this paragraph are either defined for unconstrained problems or rely on penalization-based constraint handling, which may be inefficient when dealing with a large number of constraints or when the weights are not properly tuned.

2.3.2 Approach 2: Nested optimization algorithms

A second existing approach for the optimization of VSDSP is based on using nested optimizations in order to separately handle dimensional and non-dimensional design variable. In practice, the nested formulation is comprised of an inner loop and an outer loop. For each iteration of the outer loop, which specifies the values of the dimensional variables, a complete optimization of the problem with respect to the relevant continuous and discrete variables is performed in the inner

loop. The outer loop, instead, consists of iterations of the algorithm in charge of determining the optimal dimensional variables values. The nested approach can be formulated as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \\ \text{w.r.t.} \quad & \mathbf{w} \in F_w \\ & \text{with} \quad \{\mathbf{x}^*, \mathbf{z}^*\} = \operatorname{argmin} f(\mathbf{x}, \mathbf{z}, \mathbf{w}) \\ & \qquad\qquad \text{w.r.t.} \quad \mathbf{x} \in F_x(\mathbf{w}),\ \mathbf{z} \in F_z(\mathbf{w}) \\ & \qquad\qquad \text{s.t.} \quad \mathbf{g}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \leq 0 \end{aligned} \tag{2.7}$$

This approach can be generalized by optimizing a part of the continuous and/or discrete variables within the inner loop and the remaining ones at the outer loop level. However, this might require case specific algorithms in order to properly handle the various types of variables in both optimization loops. A notable example of nested optimization of VSDSP is the the multidisciplinary optimization of an aircraft wing with the use of PSO discussed by G. Venter and J. Sobieszczanski-Sobieski in [127]. In the proposed method, the optimization of two disciplines comprising the system, structure and aerodynamics, are performed on two different levels. This allows to isolate the interactions between the two disciplines and to rely on simplified models for the analysis of their performance. At a system level, the geometry of the wing is optimized with the objective of minimizing the aerodynamic drag with respect to the number of spars, the number of ribs (dimensional variables), the type of wing cover construction (discrete variable) and the aspect and depth-to-chord ratios (continuous variables). At a sub-system level, instead, the structure of the wing is optimized with the objective of minimizing its total mass. The variables characterizing this sub-system are continuous and describe the sizing of the various structural elements. With the given separation of the problem, after every iteration of the system level optimization, the number of structural elements is defined. This iteration can then be followed by a complete optimization of the sub-system level problem in order to determine the optimal continuous sizing of the structural elements. An aircraft design problem similar to the one presented above is discussed by S. Roy in *et al.* [110], in which the simultaneous optimization of an aircraft design and its airline allocation is performed by combining and alternating a gradient-based optimization with respect to the continuous variables characterizing the problem, considering the discrete variables fixed, and an optimization with respect to the discrete variables (with respect to a surrogate model) with a B&B technique.

2.3.3 Approach 3: Sequential optimization algorithms

A third approach to the solution of VSDSP which can be identified in the literature is the sequential optimization. It is based on a similar concept as the nested optimization, with the main difference being that the optimization iterations with respect to the dimensional and non dimensional variables are performed sequentially. In practice, the algorithm alternates iterations in which the dimensional variables are considered fixed with iterations in which the continuous and discrete variables are considered fixed. This sequential approach can be schematically formulated as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{z}, \mathbf{w}^*) \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x(\mathbf{w}^*),\ \mathbf{z} \in F_z(\mathbf{w}^*) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \mathbf{z}, \mathbf{w}^*) \leq 0 \end{aligned} \quad \begin{matrix} \xrightarrow{\mathbf{x}^*, \mathbf{z}^*} \\ \xleftarrow{\mathbf{w}^*} \end{matrix} \quad \begin{aligned} \min \quad & f(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \\ \text{w.r.t.} \quad & \mathbf{w} \in F_w \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}^*, \mathbf{z}^*, \mathbf{w}) \leq 0 \end{aligned} \tag{2.8}$$

Similarly to the nested approach, a more general formulation can be defined in case both iterations of the sequential algorithm are tasked with optimizing part of the continuous and discrete variables. Notable examples of the sequential approach to VSDSP are the mesh-based optimization algorithms for mixed-variable problems presented in [4], [5], [9], [66] and [79]. Although each

paper describes a different algorithm, they share the same approach to the optimization problem which consists in an alternation between an optional user-defined search phase and a poll phase, in which an exhaustive and schematic optimum search over a mesh is performed. When applying this family of algorithms to VSDSP, it is necessary to take into account the appearance and disappearance of design variables as a function of the dimensional variables characterizing the incumbent solution around which the mesh is centered. In order to solve this issue, the mesh algorithms mentioned above alternate between searches over a mesh defined in the dimensional design space and searches over a mesh defined in the continuous and discrete search space dependent on the dimensional variables values. The application of these algorithms requires the user to provide the definition of the neighborhood of a candidate solution in the dimensional design space in order to create the mesh. The alternative solution consists in defining the mesh as the entirety of the dimensional search space, which quickly becomes unfeasible when dealing with computationally intensive problems characterized by large combinatorial design spaces. Furthermore, the user is also required to define the criteria according to which newly created continuous and discrete design variables are initialized and discarded when exploring the dimensional variable mesh. For instance, in [66] the mesh-based optimization of a thermal insulation system as a function of the number and sizing of heat intercepts is discussed. In this case, newly initialized intercepts are characterized by a technology, a cooling temperature and a thickness related to the values present within its neighborhood.

2.3.4 Approach 4: Complete exploration of the non-relaxable search space

The last and most simplistic existing approach to VSDSP consists in performing a separate and independent optimization over the relevant continuous design variables for every discrete category of the problem (*i.e.,* every possible combination of dimensional and discrete variable values) and subsequently determining the global optimum among the obtained sub-problem optima. This approach allows to fully explore the non-continuous search space and can therefore theoretically provide a consistent convergence towards the global problem optimum, depending on the continuous optimization algorithm that is considered (*i.e.,* global convergence can only be ensured under the condition that the continuous algorithm converges to the global optimum of the continuous problem). Due to the necessity of solving a separate optimization problem for every category of the considered variable-size design space problem, this approach may be inefficient, if not unfeasible, for complex problems characterized by large number of categories and/or for computationally expensive functions. The approach for the optimization of VSDSP discussed above can be formulated as follows:

$$\min \begin{cases} \min f(\mathbf{x}, \mathbf{z}_p, \mathbf{w}_q) \quad \forall \mathbf{w}_q \in F_w, \ \forall \mathbf{z}_p \in F_z(\mathbf{w}_q) \\ \text{w.r.t.} \quad \mathbf{x} \in F_x(\mathbf{w}_q) \\ \text{s.t.} \quad \mathbf{g}(\mathbf{x}, \mathbf{z}_p, \mathbf{w}_q) \leq 0 \end{cases} \tag{2.9}$$

A notable example of this approach to mixed-variable problems is developed by C.P. Frank in the context of launcher design [44]. The first step of this approach consists in analyzing and listing all of the incompatibilities between various technological choices in order to discard all non-feasible configurations and by extension limit the number of cases to optimize. Subsequently, a finite number of families of configurations, or architectures, characterized by the same design variables is determined through problem specific considerations and expertise. These configurations are then optimized in parallel with the help of a Non-dominated Sorting Genetic Algorithm NSGA-II [27] with respect to three objective functions representing the cost, the lift-off weight and the risk level related the selected technologies. Finally, the Pareto front of each architecture are merged into a single surface, thus allowing to easily compare their respective performance. The number of generations to be assigned to each architecture during a given iteration is determined as a function

of its relative performance with respect to the other architectures, thus prioritizing the most promising ones. This approach allows to fully explore the relevant search space of an otherwise computationally expensive problem and to easily compare very different system configurations. Furthermore, once the configurations to be analyzed are determined, the optimization relies on a standard mixed integer-continuous NSGA-II, which makes the implementation of the optimizer fairly straightforward. On the other hand, it is important to note that the design variables selected by the author describe the launcher at a global system level. These choices, coupled with the specific approach, allow to group together fairly different configurations which can nonetheless be described by the same variables. However, this also results in an impossibility of modeling in detail and optimize the specific different subsystems of the possible configurations. For instance, characterizing the propulsion system with high-level variables such as thrust and combustion chamber pressure, allows to regroup configurations using both solid and liquid propulsion but makes it impossible to describe sub-system specific characteristics such as the solid propulsion propellant grain geometry. A trade-off between the combinatorial size of the optimization and the level of detail with which the problem is analyzed is therefore required in case this approach is to be adapted to other applications.

2.4 Literature review synthesis

In the previous paragraphs, a number of different algorithms allowing to solve mixed-variable and VSDSP problems is presented and discussed. First of all, it can be noticed that a large number of approaches are proposed within the context of a specific optimization problem, such as launch vehicle [44] or a aircraft wing design [127]. These approaches tend to rely on the specific nature of the considered problem, like the relation between the number of spars and ribs of an aircraft wing and the associated number of wing sizing parameters, and it might therefore be challenging to generalize their working principle for any VSDSP. Furthermore, part of the discussed algorithms, such as the mesh-based optimization algorithms [4], [5], [9], [79], require the user to initialize the algorithm at a location in a search space which is not excessively far from the global optimum in order to ensure a reasonable convergence speed. In most cases, this is not feasible without user defined problem specific knowledge. Similarly, part of the discussed algorithms require the user to be able to define the neighborhood of a candidate solution. In this case as well, this is not possible when dealing with the presence of unordered discrete and dimensional variables, unless problem specific knowledge is included in the optimization process. It can also be noticed that a large portion of the presented algorithms handles constraints by penalization (or direct discarding of non feasible solutions). In general, this constraint handling technique requires a precise weight tuning in order to be efficient, which can be considerably time-consuming. It is also worth mentioning that none of the VSDSP optimization algorithms discussed above directly addresses the issue of having the feasibility domain vary from candidate solution to candidate solution as a function of the dimensional variables values. Finally, the most important drawback, common to all the methods presented in the previous paragraphs, is the total computational cost of the optimization. Indeed, both heuristic and mesh-based algorithms tend to require a large number of function evaluations in order to converge due to their necessity of exploring a large part of the design space. For most presented algorithms, this issue tends to be further accentuated when dealing with VSDSP which present a large discrete/dimensional combinatorial design space. As a consequence, within the context of computationally intensive design problems which is considered in this thesis, none of the algorithms presented in the previous paragraphs seems to provide a sufficiently fast convergence (in terms of function evaluations) to the considered problem optimum allowing to find the problem solution with an acceptable amount of computational resources. For these reasons, optimization methods relying on computationally cheap surrogate models of the considered problem objective and constraint functions are considered in the following chapters.

2.5 Conclusions

In this chapter, the optimization problem resulting from the inclusion of technological and architectural choices within the system design framework is discussed. It is shown that in the most general case, technological choices, under the form of dimensional variables, can modify the search space as a function of their values, in terms of number and type of design variables the problem functions depend on. In the same way, they can also influence the constraints a given candidate solution is subject to, and by consequence the feasibility domain. Subsequently, the existing optimization algorithms allowing to optimize the resulting variable-size design space problems are described and grouped as a function of the approach they rely on to deal with the challenges which these problems present. Among other drawbacks, this literature review shows that when considering the context of computationally intensive system design problems, the existing methods do not seem to provide a sufficiently fast convergence (in terms of function evaluations) towards the global optimum and are therefore not a viable solution.

For this reason, the approach that is followed in this work is to develop optimization methods for VSDSP and Mixed-variable problems relying on the use of surrogate models of the objective and constraint functions, also known as Surrogate Model-Based Design Optimization (SMBDO). This type of approach can usually reduce the amount of function evaluations required in order to find the problem optimum by efficiently determining the locations at which new evaluations must be performed. In Chapter 3, the particular type of surrogate modeling method considered in this thesis, known as Gaussian Process, is discussed and its extension to the mixed-variable case is detailed. In Chapter 4, a SMBDO algorithm based on said surrogate modeling technique for fixed-size mixed-variable problems is proposed. Finally, in Chapter 5 this algorithm is extended to the context of VSDSP.

CHAPTER 3

Mixed-variable Gaussian Processes

3.1 Introduction

Nowadays, a growing majority of engineering design processes heavily relies on the use of computer models and simulations, as it usually represents a faster and cheaper alternative to physical testing, while still providing results accurate enough for most applications. Furthermore, computer simulations can also provide performance estimates in conditions which cannot reasonably be tested (*e.g.,* outer space behaviors). However, computer modeling also presents a number of limitations and drawbacks. Most notably, the computation time associated to the performance simulation of complex systems can be considerably large. Typical examples involve computational fluid-dynamics analyses and finite element models. This issue can become particularly problematic when the models are used within an optimization framework, as they usually must be called a large amount of times in order to determine the solution of the considered problem (*i.e.,* optimal design), as is discussed in Chapter 2. In order to partially avoid this issue, a common solution consists in creating a surrogate model of the numerical simulation codes [105], which relies on a mathematical representation of the studied function. These surrogate models usually present a negligible computational cost when compared to the modeled functions, which allows them, if necessary, to be evaluated a large amount of times during the optimization process. However, the addition of a modeling layer also implies a loss of accuracy (*i.e.,* introduction of additional modeling errors) if compared to the actual simulation, the magnitude of which depends on the type of modeling technique that is used as well as on its parameterization.

Among the most popular surrogate modeling techniques used within the framework of complex system design optimization, one may find polynomial regression models [32], Artificial Neural Networks [95], Radial Basis Functions [34], [36], Support Vector Machine Regression [118] and Multivariate adaptive regression spline [46]. More comprehensive discussions on the differences and advantages of the various surrogate modeling techniques can be found in [105] and [128]. As mentioned in the conclusion of Chapter 2, the main focus of this thesis is towards optimization methods which rely on the use of specific modeling techniques known as Gaussian Processes [107] (GP), which are increasingly popular methods when modeling functions with low amount of available data and/or when dealing with computationally intensive optimization problems.

Most of the surrogate modeling techniques mentioned in the previous paragraph have originally been developed in order to model continuous problems (*i.e.,* characterized by functions depending solely on continuous variables). When dealing with mixed continuous/discrete functions, a number of additional challenges, such as the absence of metrics in the discrete search space, need to be addressed in order to be able to create an accurate and reliable surrogate model. In the recent years, a few adaptations of existing continuous surrogate modeling techniques as well as novel

modeling methods have been developed in order to be able to model mixed continuous/discrete functions. A few reviews of existing mixed continuous/discrete surrogate modeling techniques can be found in [16], [87] and [125]. Please note that in this chapter, the dependence of the modeled functions on dimensional variables ($\mathbf{w}$), as described in Eq. 2.1, is not considered. In practice, the surrogate modeling techniques discussed in this chapter allow to model functions which present a fixed-sized search space and only depend on continuous variables ($\mathbf{x}$) and discrete variables ($\mathbf{z}$).

The most notable surrogate modeling techniques which have been adapted and/or extended in order to enable the modeling of mixed-variable functions are discussed in the following paragraphs. All of the presented surrogate models are created by processing a training data set $\mathcal{D}$ of n samples $\{\mathbf{x}^i, \mathbf{z}^i, y^i\}$ with $i \in \{1, ..., n\}$. $\mathcal{D}$ can be defined as follows:

$$\mathcal{D} = \left\{ \mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^n\} \in F_x, \quad \mathbf{Z} = \{\mathbf{z}^1, ..., \mathbf{z}^n\} \in F_z, \quad \mathbf{y} = \{y^1, ..., y^n\} \in F_y \right\}$$

where $\mathbf{X}$ and $\mathbf{Z}$ are the matrices containing the n continuous and discrete vectors characterizing the training data set, while $\mathbf{y}$ is the vector containing the associated responses (*i.e.,* modeled function). F_x, F_z and F_y are the definition domains of the 3 previously mentioned matrices.

Radial Basis Function

The Radial Basis Function (RBF) surrogate modeling technique, originally proposed by Hardy [56], consists in defining the surrogate model as a sum of distance-based basis functions $\phi(\cdot)$ centered around the n samples of the considered data set:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n} \theta^i \phi \left(||\mathbf{x} - \mathbf{x}^i|| \right) \tag{3.1}$$

Thanks to the fact that the RBF only depends on the distance between samples, it can easily be extended in order to model mixed continuous/integer functions (sometimes referred to as mixed-integer) by computing the Euclidean distance between samples directly in the continuous/integer design space. This approach is applied for the solution of mixed-integer constrained problems in a few papers [60], [73], [91], [106], where the computationally intensive objective and/or constraint functions are replaced by RBF based models. These models are then used in order to determine the promising areas of the search space by either directly optimizing the surrogate model of the considered problem with the help of standard algorithms such as Mixed-integer evolution strategies [73] and Mixed-integer nonlinear programming [106], or by relying on alternative criteria such as the probability of improvement [60] and constraint augmented criteria [91]. In [90], [132] and [133], the possibility of modeling mixed-variable functions by relying on continuous surrogate models, such as RBF and linear models by replacing the use of Euclidean distance with a alternative definition is considered. The Hamming distance [81] as well as swap and interchange distances based on the permutations between discrete variable values are considered. An RBF assisted mixed-variable GA is discussed in [11], in which the GA performs the objective function optimization with respect to the surrogate model rather than on the computationally intensive functions. At each iteration, the actual function value at the surrogate model optimum location is computed and added to the model data set. In this case, the discrete variables are handled by grouping the training data in clusters with respect to the their value and subsequently defining a separate surrogate model for each cluster. The distance between clusters is then computed through the use of the Hamming distance. Similarly, a coding-based RBF-assisted mixed-variable GA is also proposed in [17] for the optimization of a Hyper-sonic cryogenic tank design, however, due to the proprietary nature of the used software no detailed information on the implementation is known.

Support Vector Machine

In its original formulation, Support Vector Machine (SVM) is a linear classification algorithm allowing to characterize the hyperplane dividing the points of a purely continuous data set in 2 distinct groups with the highest margin [118]. This hyperplane is defined in such a way that:

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x}+b>0 \qquad & \forall y_i = 1 \\ \mathbf{w}\cdot\mathbf{x}+b<0 \qquad & \forall y_i = -1 \end{aligned} \tag{3.2}$$

where $\mathbf{w}$ and b are the hyperparameters of the model which need to be tuned in order for the model to properly classify the data. SVM can be used in order to perform non-linear classification by replacing the dot product with a kernel function $k(\mathbf{x},\mathbf{x}') = \langle\phi(\mathbf{x}),\phi(\mathbf{x}')\rangle$ [119], such as a squared-exponential kernel: $k(\mathbf{x},\mathbf{x}') = \exp(-\theta||\mathbf{x}-\mathbf{x}'||^2)$. SVM can be further extended in order to perform regression (*i.e.,* surrogate modeling) by considering the hyperplane as the modeled function prediction:

$$\hat{y}(\mathbf{x}) = \mathbf{w}\cdot\phi(\mathbf{x}) + b \tag{3.3}$$

In this case, the hyperparameters are optimized in such a way to ensure that the modeling error is bounded by a given threshold ϵ while also minimizing the model complexity. In order to extend the application of SVM regression to mixed-variable functions, Herrera *et al.,* [59] suggest mapping the l levels of a discrete variable through dummy coding onto a $l-1$ auxiliary binary variables. As a result, all the levels of a given discrete variable are equidistant in this auxiliary space. Furthermore, in order to better handle the inherent structural differences between continuous and discrete variables, the SVM algorithm is coupled with the use of multiple kernel regression [25]. In other words, the global kernel is defined as a weighted sum of various kernels, thus allowing to better capture the trends related to the discrete variables.

Generalized Linear Models

Generalized Linear Models (GLM) [85] can be seen as a stochastic generalization of linear models. The prediction of GLM for purely continuous functions is defined as a distribution of the exponential family (*e.g.,* normal, gamma) with a given mean $\hat{y}(\mathbf{x})$ and constant variance σ^2. Furthermore, the prediction mean is related to a linear predictor η through a so-called link function $g(\cdot)$ in the following fashion:

$$\eta = \sum_{i=1}^{n} \mathbf{x}^i\beta_i = g(\hat{\mathbf{y}}) \tag{3.4}$$

where $\beta_i,\ldots,\beta_n$ are the hyperparameters of the model. In order to apply GLM to model mixed-variable functions, the clustering based mixed-variable RBF proposed in [10] and [11] is combined with a GLM fitting with respect to the discrete variables represented through coding and the residual of the continuous RBF model. The global surrogate model is finally computed as the sum between the RBF and the GLM predictions.

Spline regression

Spline regression is a non parametric surrogate model in which the prediction of the modeled function is computed as a conditional mean $\hat{y}(\mathbf{x})$ and the associated variance $\hat{\sigma}(\mathbf{x})$. In order to deal with the dependence of the modeled function on both continuous and discrete variables, Ma *et al.* [80] suggest defining the conditional mean by weighting the continuous tensor-product polynomial splines with the help of categorical kernel functions:

$$\hat{y}(\mathbf{x},\mathbf{z}) = \mathcal{B}(\mathbf{x})\beta(\mathbf{z}) \tag{3.5}$$

where $\mathcal{B}$ are the continuous tensor-product polynomial splines and β is a vector containing the weights associated to the discrete kernels presented in [72].

Moving least squares

The Moving Least Squared (MLS) surrogate modeling technique [68] can be seen as a generalization of the least squares method. The prediction of purely continuous modeled functions is computed as follows [22]:

$$\hat{y}(\mathbf{x}) = \mathbf{p}(\mathbf{x})^T \mathbf{a}(\mathbf{x}) \tag{3.6}$$

$\mathbf{p}(\mathbf{x})$ represents a polynomial basis (*e.g.*, $[1,\ x_1,\ x_2,\ x_1^2,\ x_1x_2, x_2^2, \dots]$) while $\mathbf{a}(\mathbf{x})$ is computed by minimizing the following expression:

$$\mathbf{a}(\mathbf{x}) \ = \underset{\text{w.r.t. } \mathbf{a}}{\text{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}^i, \mathbf{x}) \left(p(\mathbf{x}^i)^T \mathbf{a} - \mathbf{y} \right)^2 \right) \tag{3.7}$$

where w_i are weights computed as a function of the distance between $\mathbf{x}$ and $\mathbf{x}^i$. Similarly to what is proposed in [59] for the SVM, the mixed-variable adaptation for MLS which is proposed by Coelho in [37] relies on mapping the l levels of a discrete variable through dummy coding onto a $l-1$ auxiliary binary variables, thus obtaining a simplex on the auxiliary space. Also in this case, the inherently different nature of the original discrete variable must be taken into account when defining the way the weights w_i are computed.

Artificial neural networks

Artificial Neural Networks (ANN) are a class of regression and categorization methods [131], [57], which have become increasingly popular in the past years thanks to the availability of a larger processing power. The underlying principle of ANN is inspired from the functioning of the brain and relies on computing the model prediction as a combination of subsequent non-linear functions of the input. In practice, this input (or signal) is processed by several layers, each one containing a given number of nodes (*i.e.,* non-linear functions). The output of each node is computed through a non-linear function of the sum of the signals it receives from other nodes, usually of all of the previous layer nodes. In most ANN implementations, the node functions as well as the connections between the nodes are characterized by different weights which must be optimized in order for the network to provide accurate predictions.

Different variants of ANN with specific characteristics and/or training approaches exist, such as feed-forward neural networks [113], convolutional neural networks and generative adversarial networks [49]. However, similarly to the other surrogate modeling techniques previously discussed, most of the ANN variants are developed for purely continuous inputs, while only a few have been adapted in order to handle the presence of discrete variables. For instance, in [38] the impact of relying on a coding approach in order to perform the classification of mixed-variable data with standard continuous/integer ANN is discussed. The results show that the more suitable type of encoding depends on the specific characteristics of the considered problem, and on whether ordinal or nominal discrete variables are considered. A similar coding-based approach is used in [120] to model system responses in an MDO framework, in which both one hot and binary encodings are considered and compared. In this case as well, a difference in modeling performance between the considered encodings depending on the modeled function characteristics is shown. In [23], an alternative approach to encoding is proposed under the form of a feed-forward Multi-Layer Perceptron (MLP). The underlying idea is to define a multi-outoput MLP with respect to the continuous variables the considered function depends on, with a number of outputs equivalent to the number of categories characterizing the discrete variables. The resulting prediction is then associated to the output corresponding to the unmapped sample category. In [77], an Adaptive-Network-based Fuzzy Inference System (ANFIS) variant is proposed for the modeling of mixed-variable functions. The adaptation relies on processing the continuous variables with a standard ANFIS approach, whereas the discrete variables are independently

transformed through the help of a one hot encoding and multiplied by a parameterized firing-strength transform matrix characterizing the contributions of the various discrete variable levels to the global output. The resulting signal is then directly forwarded to the main ANFIS framework in order to determine the weight associated to the nodes of the last network layer. A last notable example of ANN adaptation for mixed-variable problems is represented by the generalization of Self-Organizing Maps (SOM) for categorical data presented in [61]. The main idea of the presented approach consists in relying on a concept of tree-like hierarchical distance defined in the mixed continuous/discrete search space. In practice, the search space is decomposed into hierarchically dependent higher and lower level concepts which are interconnected by weighted links. The resulting continuous/discrete distance, computed as the total link weight between the two considered leaf nodes, can then be used in order to perform SOM-based categorization with respect to mixed-variable data. Please note that due to the large number of ANN variants and applications, the list of mixed-variable adaptations presented above is not exhaustive. A more comprehensive discussion on the topic can be found in [47]. However, it is also important to note that ANN tend to require large data sets in order to properly tune the hyperparameters of the model, which is not consistent with the computationally intensive design framework considered in this thesis.

This chapter focuses on the adaptations of Gaussian Process based surrogate modeling in order to perform the modeling of fixed-sized functions which depend simultaneously on continuous and discrete variables. GP are increasingly popular methods when dealing with computationally intensive functions due to their good modeling performance and flexibility. They present several advantages when compared to other surrogate modeling techniques. For instance, the training of the hyperparameters can be directly performed with respect to the training data, and does not require auxiliary data sets, as is for instance the case when using cross-validation. Additionally, GP can be parameterized in order to automatically handle noisy training data sets, which can for example be encountered when considering experimental data. Furthermore, although they do not explicitly require it, GP allow the user to easily include problem specific knowledge in the model definition, if available (*e.g.,* specification of the mean function and selection of the kernel). Finally, and most importantly, GP can return an estimated error associated to prediction of the modeled function as a (virtually) computationally free bi-product, under the form of a variance value. As is discussed in Chapter 4, this particular characteristic can be useful when performing Surrogate Model-Based Design Optimization, as it allows to define surrogate model refinement criteria characterized by a trade-off between exploitation (*i.e.,* refinement of the incumbent solution) and exploration (*i.e.,* reduction of the model error and uncertainty) of the design space.

The main objective of this chapter is to provide a comprehensive discussion on the necessary steps required to model mixed-variable functions with the help of GP as well as present the alternative adaptations under the same formalism in order to better highlight their resulting strengths and weaknesses. Following this introduction, in the second Section a theoretical overview of Gaussian Processes is provided. In the third Section, the construction of valid discrete variable kernels is discussed, and the existing kernels are re-defined within this formalism, thus allowing to better highlight and discuss the differences between the presented kernels. In the fourth Section, the discrete kernels are compared from a performance perspective by testing their modeling capabilities on a number of analyitical and engineering related test-cases of varying complexity. In the fifth Section the obtained results are described and discussed, and finally the relevant conclusions are provided in the sixth and last Section. Finally, please note that this chapter is an extension of the work presented in a book chapter by Pelamatti *et al.,* [98].

3.2 Gaussian Process surrogate models

The core concept of Gaussian Process based surrogate modeling (sometimes also referred to as Kriging [94], [111]) is to predict the response value y^* of a black-box function $f(\cdot)$ for a generic unmapped input $\{\mathbf{x}^*, \mathbf{z}^*\}$ through an inductive procedure. Generally speaking, the larger the training data set is, the more accurate the model will be. However, it is important to note that the choice of samples that are included in the training set also influences the modeling accuracy, as is discussed later in the chapter. In its most generic definition, a Gaussian Process is *a collection of random variables, any finite number of which have a joint Gaussian distribution*, or alternatively *is a generalization of the Gaussian probability distribution* [107]. In other words, instead of describing the probability distribution of random scalar or vectorial variables, GP map the probability distribution of the possible regression functions. A generic Gaussian Process $Y(\mathbf{x}, \mathbf{z})$ is characterized by its mean function μ:

$$\mu(\mathbf{x}, \mathbf{z}) = E[Y(\mathbf{x}, \mathbf{z})] \tag{3.8}$$

and its covariance function:

$$Cov\left(Y(\mathbf{x}, \mathbf{z}), Y(\mathbf{x}', \mathbf{z}')\right) = E[(Y(\mathbf{x}, \mathbf{z}) - \mu(\mathbf{x}, \mathbf{z}))(Y(\mathbf{x}', \mathbf{z}') - \mu(\mathbf{x}', \mathbf{z}'))] \tag{3.9}$$

and if a generic function $f(\cdot)$ follows a GP, it can be expressed as:

$$f \sim GP(\mu(\cdot), Cov(\cdot)) \tag{3.10}$$

The mean function parameterization can technically be defined by the user in order to better represent the modeled function. A popular choice of parameterization, thanks to its flexibility, is the polynomial function. However, in most real-life engineering design cases, insufficient information regarding the global trend of the modeled functions is known, and it is therefore complicated to choose the appropriate trend and relative parameterization of $\mu(\cdot)$. In these cases, it is common practice to consider the regression function μ as a being constant with respect to the design space [117]:

$$\mu(\mathbf{x}, \mathbf{z}) = \mu \tag{3.11}$$

The prior mean and prior covariance are updated by relying on the information on the modeled function provided by the data set $\mathcal{D}$, which enables to provide a more insightful model of the considered function. The predicted value f^* of this function at an unmapped location $\{\mathbf{x}^*, \mathbf{z}^*\}$ is then computed under the form of a Gaussian distribution conditioned on the data set [107]:

$$f^*|\mathbf{x}^*, \mathbf{z}^*, \mathbf{X}, \mathbf{Z}, \mathbf{Y} \sim \mathcal{N}\left(\hat{y}(\mathbf{x}^*, \mathbf{z}^*), \hat{s}^2(\mathbf{x}^*, \mathbf{z}^*)\right) \tag{3.12}$$

In other words, the GP provides the predicted value of the modeled function at an unmapped location $\{\mathbf{x}^*, \mathbf{z}^*\}$ under the form of a mean value $\hat{y}(\mathbf{x}^*, \mathbf{z}^*)$:

$$\begin{aligned} \hat{y}(\mathbf{x}^*, \mathbf{z}^*) &= \mathbb{E}[f^*|\mathbf{x}^*, \mathbf{z}^*, \mathbf{X}, \mathbf{Z}, \mathbf{Y}] \\ &= \mu + Cov(Y(\mathbf{x}^*, \mathbf{z}^*), Y(\mathbf{X}, \mathbf{Z}))Cov(Y(\mathbf{X}, \mathbf{Z}), Y(\mathbf{X}, \mathbf{Z}))^{-1}(\mathbf{Y} - \mu) \\ &= \mu + \boldsymbol{\psi}^T(\mathbf{x}^*, \mathbf{z}^*)\mathbf{K}^{-1}(\mathbf{y} - \mathbf{1}\mu) \end{aligned} \tag{3.13}$$

and associated variance $\hat{s}^2(\mathbf{x}^*, \mathbf{z}^*)$:

$$\begin{aligned} \hat{s}^2(\mathbf{x}^*, \mathbf{z}^*) &= Var(f^*|\mathbf{x}^*, \mathbf{z}^*, \mathbf{X}, \mathbf{Z}, \mathbf{Y}) \\ &= Cov(Y(\mathbf{x}^*, \mathbf{z}^*), Y(\mathbf{x}^*, \mathbf{z}^*)) - \\ &\quad Cov(Y(\mathbf{x}^*, \mathbf{z}^*), Y(\mathbf{X}, \mathbf{Z}))\ Cov(Y(\mathbf{X}, \mathbf{Z}), Y(\mathbf{X}, \mathbf{Z}))^{-1}\ Cov(Y(\mathbf{X}, \mathbf{Z}), Y(\mathbf{x}^*, \mathbf{z}^*)) \\ &= k(\{\mathbf{x}^*, \mathbf{z}^*\}, \{\mathbf{x}^*, \mathbf{z}^*\}) - \boldsymbol{\psi}^T(\mathbf{x}^*, \mathbf{z}^*)\mathbf{K}^{-1}\boldsymbol{\psi}(\mathbf{x}^*, \mathbf{z}^*) \end{aligned} \tag{3.14}$$

where $\mathbf{K}$ is the $n \times n$ Gram covariance matrix containing the covariance values between the n sample of the data set:

$$\mathbf{K}_{i,j} = Cov\left(Y(\mathbf{x},\mathbf{z}), Y(\mathbf{x}',\mathbf{z}')\right) = k(\{\mathbf{x},\mathbf{z}\},\{\mathbf{x}',\mathbf{z}'\}) \tag{3.15}$$

$\mathbf{y}$ is a $n \times 1$ vector containing the responses corresponding to the n data samples:

$$y_i = f(\mathbf{x}^i,\mathbf{z}^i) \quad \text{for } i = 1,\dots,n \tag{3.16}$$

$\mathbf{1}$ is a $n \times 1$ vector of ones and finally $\boldsymbol{\psi}$ is an $n \times 1$ vector containing the covariance values between each sample of the training data set and the point at which the function is predicted:

$$\boldsymbol{\psi}_i(\mathbf{x}^*,\mathbf{z}^*) = Cov(Y(\mathbf{x}^*,\mathbf{z}^*), Y(\mathbf{x}^i,\mathbf{z}^i)) = k(\{\mathbf{x}^*,\mathbf{z}^*\},\{\mathbf{x}^i,\mathbf{z}^i\}) \qquad \text{for } i = 1,\dots,n \tag{3.17}$$

Within the GP framework, the covariance function is defined as an input space dependent parameterized function, as is discussed later on in Section 3.4.

As an illustrative example, a simple purely continuous one-dimensional function $f(\cdot)$ is considered:

$$f(x) = \cos(2\pi x) + 1.5x^2 \tag{3.18}$$

Supposing that no information on the modeled function is known, the prior mean is set to be a constant:

$$\mu = 0 \tag{3.19}$$

while the covariance prior is defined as a squared exponential function, as is discussed in Section 3.4.:

$$k(x,x') = \exp(\theta|x - x'|^2) \tag{3.20}$$

If no training data is provided to the model, predictions can be sampled from the prior, as is shown in Figure 3.1. As can be expected, the predicted functions are completely unrelated to the modeled function, due to the fact that no information is provided to the model. However, if training data

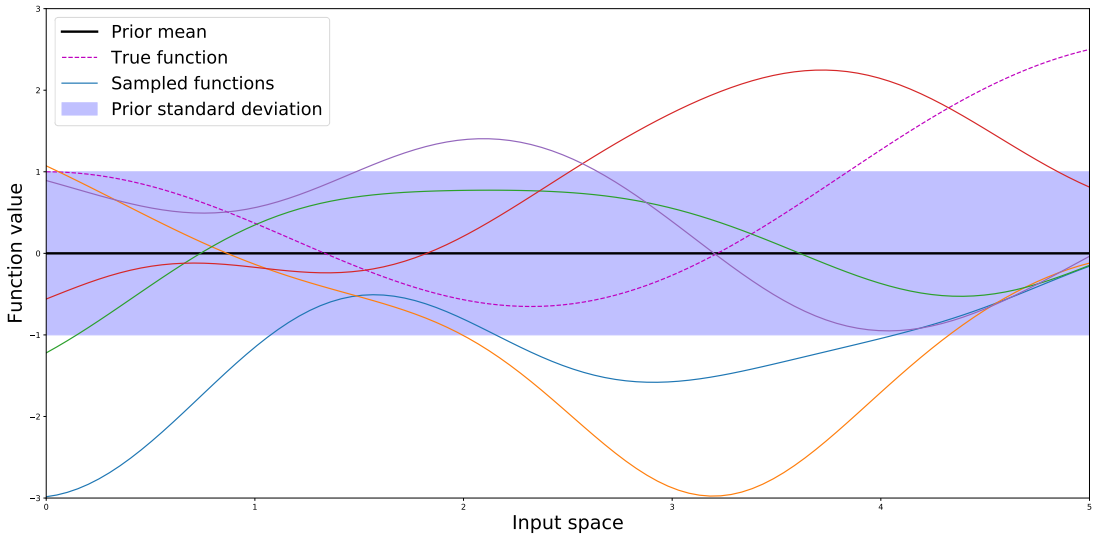


FIGURE 3.1: Example of a Gaussian Process prior modeling

samples (*i.e.*, evaluations of the modeled function) are provided to the GP model, the prior can be updated by relying on this new information, thus obtaining the posterior prediction of the modeled function. An example of posterior prediction by relying on 4 data samples is illustrated in Figure 3.2. It can be noticed that in the posterior prediction, both the sampled functions and

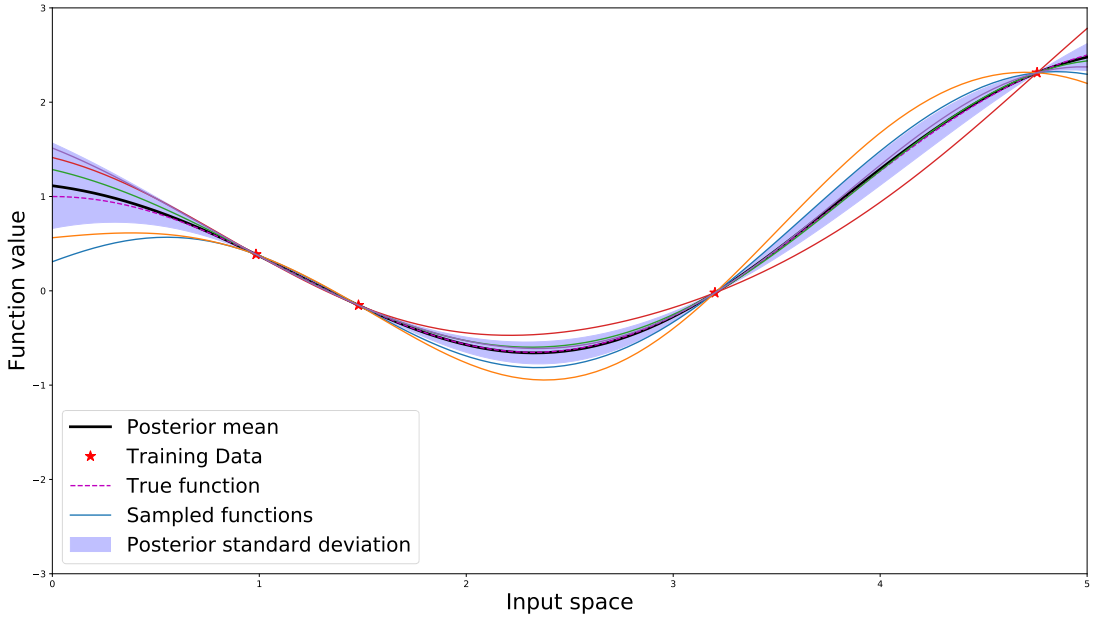


FIGURE 3.2: Example of a Gaussian Process posterior modeling

the posterior mean match very closely the modeled function. Moreover, the posterior standard deviation is also considerably reduced when compared to the prior. As previously mentioned, a GP prediction is usually provided in term of posterior mean $\hat{y}$ and associated posterior variance $\hat{s}^2$, as sampling functions over the entire search space is more complex, and not necessary within the optimization framework. Finally, Eqs. 3.14 and 3.15 show that the variance $s^2(\cdot)$ associated to a given prediction $\hat{y}(\cdot)$ can be computed at a very limited computational cost, thanks to the fact that they both share the same computationally costly term of the prediction equation: $\boldsymbol{\psi}^T(\mathbf{x}^*, \mathbf{z}^*)\mathbf{K}^{-1}$. This property can be particularly useful within the optimization framework, as is discussed in Chapter 4.

3.3 Mixed-variable Gaussian Process modeling

As mentioned in the introduction, there are relatively few techniques allowing to model functions which depend simultaneously on continuous and discrete variables. Reviews and comparisons of the existing techniques can be found in [125] and [136]. The most commonly used approach when dealing with this kind of functions consists in creating a separate and independent GP model for every category of the considered problem by relying solely on the training data relative to said category. In the remainder of this work, this approach is referred to as **Category-wise surrogate modeling**. However, within the framework of computationally intensive design it is often unfeasible to provide the amount of data for each category of the problem necessary to model the considered function accurately enough for optimization purposes. This issue becomes particularly relevant when dealing with problems characterized by a large number of categories [125]. For this reason, in this thesis the concept of mixed-variable surrogate modeling is explored. The underlying idea is to maximize the use of the information provided by the training data set by creating a single mixed continuous/discrete GP with the entirety of the available data rather than distributing the available information over several independent continuous surrogate models. For illustrative purposes, the following simple mixed variable function $f(\cdot)$ is considered:

$$\begin{aligned} &f(x,z) = \cos(x) + 0.5z \\ &x \in [0,7],\ z \in \{0,1\} \end{aligned} \tag{3.21}$$

If few data samples are available in order to model the function, as is shown in Figure 3.3, the prediction provided by the separate continuous GP defined with respect to the two categories of the considered function might be highly inaccurate, as is shown in the top half of Figure 3.4.

If instead the exact same data samples are used in order to create a mixed-variable GP, the

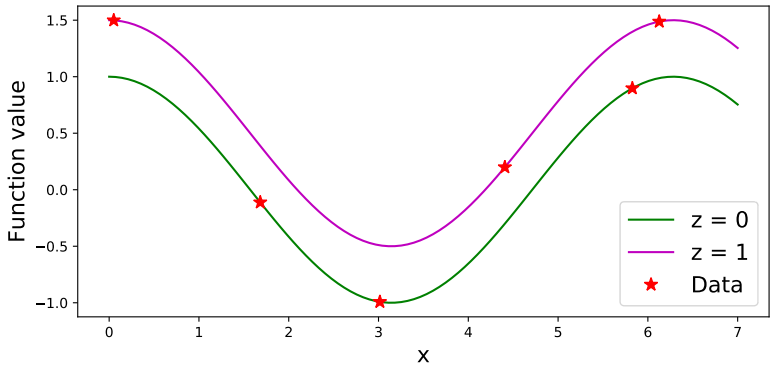


FIGURE 3.3: Mixed-variable modeling example function

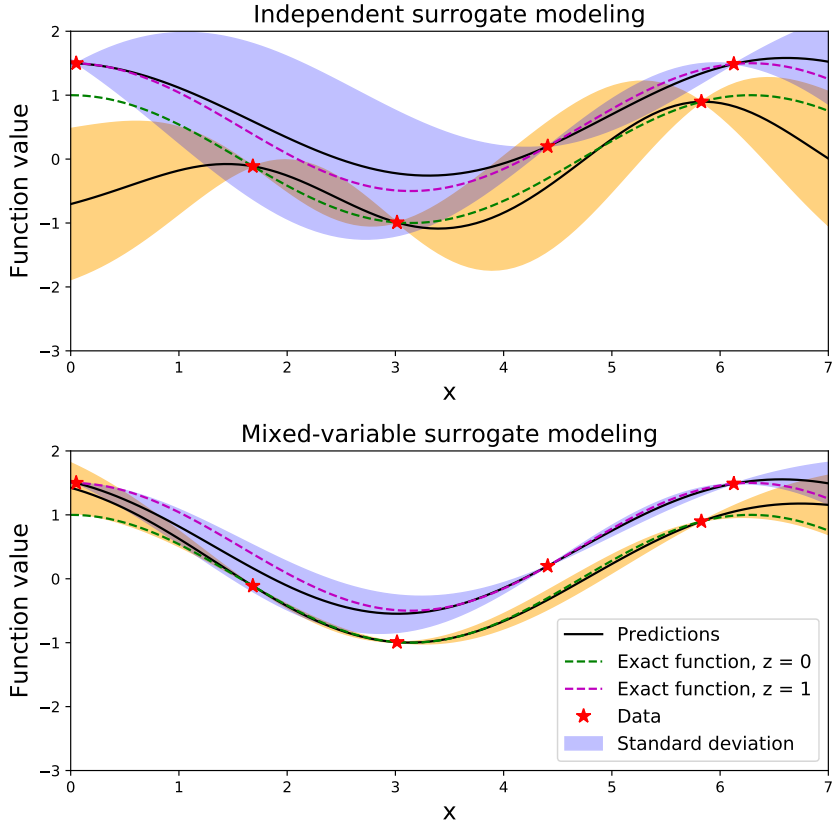


FIGURE 3.4: Comparison between independent category-wise modeling and mixed-variable modeling

prediction is considerably more accurate over the entirety of the search space, while the associated variance is also reduced, as is shown in the lower part of Figure 3.4. By relying on mixed-variable surrogate modeling rather than category-wise modeling, it is therefore possible to better exploit the information provided by the data samples defined in the mixed-variable search space, thus improving the modeling accuracy. Within the context of surrogate model-based optimization,

a higher GP modeling accuracy tends to result in a faster convergence rate, thus reducing the computational cost of the optimization process.

Please note that an alternative approach for the inclusion of discrete variables within the framework of GP based modeling is represented by the so-called Treed Gaussian Processes (TGP). The underlying idea of this surrogate modeling technique consists in defining a number of separate and independent continuous GP models over complementary partitions of the search space [52]. These tree-like partitions are obtained by recursively performing binary splits on the value of a single variable. The hyperparameters of the independent GP as well as the tree partition are optimized simultaneously with the help of a Markov Chain Monte Carlo. A mixed-variable adaptation of TGP is implemented in the R *mlr* package [19] and is based on a binary coding of the discrete variables the considered problem depends on. In practice, the local GP models are defined by considering the binary variables as integers, while the binary tree partitioning can naturally be performed with respect to these auxiliary binary variables. However, due to the considerably different nature of this method when compared to the single mixed-variable kernel approach, as well as the difference in the approach with which the hyperparameters are trained, TGP mixed-variable modeling is not considered in this thesis.

3.4 Gaussian Process kernels

The covariance function $k(\cdot)$ is the core component of a Gaussian Process based surrogate model [107] and is one of the main focuses of this manuscript. Loosely speaking, the purpose of this function is to characterize the similarity between distinct data samples in the design space with respect to the modeled function. In order for a function $k(\cdot)$ to represent a valid covariance, there are two main requirements [7]. More specifically, the function must be symmetric:

$$k(\{\mathbf{x},\mathbf{z}\},\{\mathbf{x}',\mathbf{z}'\}) = k(\{\mathbf{x}',\mathbf{z}'\},\{\mathbf{x},\mathbf{z}\}) \tag{3.22}$$

and positive semi-definite over the input space, *i.e.*:

$$\begin{aligned} &\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j k(\{\mathbf{x},\mathbf{z}\},\{\mathbf{x}',\mathbf{z}'\}) \geq 0 \\ &\forall n \geq 1,\ \forall (a_1,\dots,a_n) \in \mathbb{R}^n \quad \text{and} \quad \forall \{\mathbf{x},\mathbf{z}\},\{\mathbf{x}',\mathbf{z}'\} \in F_x \times F_z \end{aligned} \tag{3.23}$$

Alternatively, a positive semi-definite function can also be defined by ensuring that the $n \times n$ matrix constructed by computing each element $\mathbf{K}_{i,j}$ as:

$$\mathbf{K}_{i,j} = k(\{\mathbf{x}^i,\mathbf{z}^i\},\{\mathbf{x}^j,\mathbf{z}^j\}) \qquad \text{for } i,j = 1,\dots,n \tag{3.24}$$

is positive semi-definite:

$$\mathbf{a}^T\mathbf{K}\mathbf{a} \geq 0 \quad \forall\ \mathbf{a} \in \mathbb{R}^n \tag{3.25}$$

Thanks to the characteristics mentioned above, a valid covariance function can, by construction, be defined as a parameterizable Hilbert space kernel [107], as is discussed in the following paragraphs.

For the sake of clarity, the brief introduction to Hilbert space kernels is discussed within the purely continuous case, and is extended to the mixed-variable case in the subsequent paragraphs. Let $F_x \subseteq \mathbb{R}^{n_x}$ be a non-empty set, a kernel function on F_x, *i.e.*, $k : F_x \times F_x \to \mathbb{R}$, can be defined if there exists an $\mathbb{R}$-Hilbert space and a map $\phi : F_x \to \mathcal{H}$ such that $\forall \mathbf{x},\mathbf{x}' \in F_x$:

$$k(\mathbf{x},\mathbf{x}') := \langle \phi(\mathbf{x}),\phi(\mathbf{x}')\rangle_{\mathcal{H}} \tag{3.26}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the inner product on the Hilbert space $\mathcal{H}$. By definition, an inner product defined in a Hilbert space must be [119], [115]:

- bi-linear: $\langle \alpha_1 \phi(\mathbf{x}) + \alpha_2 \phi(\mathbf{x}'), \phi(\mathbf{x}'') \rangle_{\mathcal{H}} = \alpha_1 \langle \phi(\mathbf{x}), \phi(\mathbf{x}'') \rangle_{\mathcal{H}} + \alpha_2 \langle \phi(\mathbf{x}'), \phi(\mathbf{x}'') \rangle_{\mathcal{H}}$
- symmetric: $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}) \rangle_{\mathcal{H}}$
- positive: $\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle_{\mathcal{H}} \geq 0, \quad \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle_{\mathcal{H}} = 0$ if and only if $\phi(\mathbf{x}) = 0$

for any $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in F_x$. The Hilbert space $\mathcal{H}$ can be seen as a space onto which specific features of the considered design variables are mapped. If necessary, multiple mappings and associated kernels can be defined for the same design variable in case several distinct features of the considered variable must be taken into account. Technically, each design variable can be mapped onto an infinity of distinct Hilbert spaces in order to map infinitely different features. In order to better capture and model the dependence of the GP covariance function with respect to the various variables of the design space, both the mapping function $\phi(\cdot)$ and the Hilbert space inner product $\langle \cdot \rangle_{\mathcal{H}}$ can depend on a number parameters, known as hyperparameters [107]. The kernel hyperparameters can be tuned as a function of the available data in order to maximize the modeling accuracy of the surrogate model. The process of determining the optimal values of the hyperparameters is often referred to as the **GP training** and it is discussed later on in this chapter. Please note that in the remainder of the manuscript, the terms covariance function and covariance kernel are used interchangeably.

For illustrative purposes, the constructions of two commonly used kernels, namely the squared exponential kernel [112] (sometimes referred to as RBF kernel) and the constant kernel, are described below. For the construction of the squared exponential kernel, the input space is mapped onto an infinite power series:

$$\begin{aligned} &\phi(\mathbf{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{\infty} \\ &\phi(\mathbf{x}) = \sigma_x \exp(-\theta ||\mathbf{x}||_2^2) \left[1, \sqrt{2\theta}\mathbf{x}, \frac{2\theta \mathbf{x}^2}{2}, ..., \frac{(\sqrt{2\theta}\mathbf{x})^d}{d!} \right]^T \text{ with } d = 0, ..., \infty \end{aligned} \tag{3.27}$$

where θ and σ_x are respectively the lengthscale hyperparameter and the standard deviation associated to the kernel. The kernel is then defined as the scalar product between the power series:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \phi(\mathbf{x})^T \phi(\mathbf{x}') \\ &= \sigma_x^2 \exp(-\theta ||\mathbf{x}||_2^2) \exp(-\theta ||\mathbf{x}'||_2^2) \left(1 + 2\theta \mathbf{x}\mathbf{x}' + ... + \frac{(2\theta \mathbf{x}\mathbf{x}')^d}{d!} + ... \right) \\ &= \sigma_x^2 \exp(-\theta ||\mathbf{x}||_2^2) \exp(-\theta ||\mathbf{x}'||_2^2) \sum_{d=0}^{\infty} \frac{(2\theta \mathbf{x}\mathbf{x}')^d}{d!} \\ &= \sigma_x^2 \exp(-\theta ||\mathbf{x}||_2^2) \exp(-\theta ||\mathbf{x}'||_2^2) \exp(2\theta \mathbf{x}\mathbf{x}') \\ &= \sigma_x^2 \exp(-\theta ||\mathbf{x}||_2^2) - \theta ||\mathbf{x}'||_2^2 + 2\theta \mathbf{x}\mathbf{x}') \\ &= \sigma_x^2 \exp\left(-\theta ||\mathbf{x} - \mathbf{x}'||_2^2\right) \end{aligned} \tag{3.28}$$

Thanks to the fact that the scalar product satisfies all the requirements to be a valid inner product on $\mathbb{R}^{\infty}$, the resulting kernel is valid by construction.

A similar derivation can be performed for the constant kernel, which returns the same covariance between any pair of inputs. Let $\phi(\cdot)$ be the mapping onto $\mathcal{H} \subset \mathbb{R}$:

$$\begin{aligned} &\phi(\mathbf{x}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{+} \\ &\phi(\mathbf{x}) = \sqrt{\theta} \quad \text{with} \quad \theta \geq 0 \end{aligned} \tag{3.29}$$

where θ is the hyperparameter characterizing the constant value returned by the kernel. This kernel can then be defined as the scalar product between the two mappings:

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} = \phi(\mathbf{x})^T \phi(\mathbf{x}') = \theta \tag{3.30}$$

In the same way, a large number of alternative kernels defined in the continuous search space exist, such as the linear kernel [107]:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_x^2 \mathbf{x}\mathbf{x}' \tag{3.31}$$

the polynomial kernel of degree p [115]:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_x^2 (\mathbf{x}\mathbf{x}' + \theta_c)^{\theta_p} \tag{3.32}$$

and the Matern kernels class [88]. *e.g.,* the $\frac{5}{2}$ Matern function:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_x^2 \left(1 + \frac{\sqrt{5}||\mathbf{x} - \mathbf{x}'||_2}{\theta} + \frac{5||\mathbf{x} - \mathbf{x}'||_2^2}{3\theta^2}\right) \exp\left(-\frac{\sqrt{5}||\mathbf{x} - \mathbf{x}'||_2}{\theta}\right) \tag{3.33}$$

Depending on the characteristics of the modeled function, some specific kernel choices might be more appropriate than others. In general, a preliminary analysis of the problem at hand is necessary in order to select the kernel which can provide the most accurate and/or robust modeling. Nevertheless, the concepts presented in this work are valid and applicable regardless of the type of kernels which are chosen in order to model the influence of the continuous design variables on the considered functions.

3.4.1 Kernel operators

In complex design problems, a single kernel may not be sufficient in order to capture the different influences of the various design variables. The main reason for this limitation, is that the same single set of hyperparameters is used to characterize the covariance between every dimension of the compared samples. For this reason, it is common practice to combine kernels defined over various sub-spaces of the design space, thus resulting in a valid kernel defined over the entire search space which provides a more accurate modeling of the considered function. It can be shown that kernels can be combined while still resulting in a valid covariance function, as long as the chosen operator allows it [119]. In this thesis, the 3 following kernel operators are considered:

- **Sum**

  Let $k_1(\cdot)$ be a kernel defined on the input space F_{x_1} and $k_2(\cdot)$ be a kernel defined on F_{x_2}. It can be shown that $k(\cdot) = k_1(\cdot) + k_2(\cdot)$ is a valid kernel on the input space $F_x = F_{x_1} \times F_{x_2}$.

- **Product**

  Let $k_1(\cdot)$ be a kernel defined on the input space F_{x_1} and $k_2(\cdot)$ be a kernel defined on F_{x_2}. It can be shown that $k(\cdot) = k_1(\cdot) \times k_2(\cdot)$ is a valid kernel on the input space $F_x = F_{x_1} \times F_{x_2}$. Furthermore, let $k(\cdot)$ be a kernel defined on the input space F_x and $\alpha \in \mathbb{R}^+$, $\alpha k(\cdot)$ is also a valid kernel on F_x.

- **Mapping**

  Let $k(\cdot)$ be a kernel on F_x, let $\tilde{F}_x$ be a set and let $A : \tilde{F}_x \to F_x$ be a mapping function. Then $\tilde{k}(\cdot)$ defined as $\tilde{k}(\mathbf{x}, \mathbf{x}') := k(A(\mathbf{x}), A(\mathbf{x}'))$ is a kernel on $\tilde{F}_x$

The kernel operators can be used in order to combine kernels in various fashions. Two popular examples are the ANOVA and the exponential kernel [112]:

- ANOVA: $k(\{x_1, x_2\}, \{x_1', x_2'\}) = (1 + k_{x_1}(x_1, x_1')) \times (1 + k_{x_2}(x_2, x_2'))$
- Exponential: $\exp(k(x, x')) = \sum_{n=0}^{n=\infty} \frac{k(x,x')^n}{n!}$

The combination of kernels has 2 main advantages: first, it enables defining distinct kernels over different design variables, which allows to better capture and model the influence the various design variables on the modeled function. This is particularly important when dealing with problems in which different variables are related to considerably different trends of the modeled function, as often happens within the framework of system design. Furthermore, different kernels can also be combined in order to model more accurately different trends characterized by the same design variable or group of variables.

A popular approach when dealing with multidimensional problems consists in defining a single separate kernel $k_d(\cdot)$ for each dimension d the considered problem depends on, and computing the global kernel as the product between one-dimensional kernels [112]:

$$k(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{n_x} k_d(x_d, x_d') \tag{3.34}$$

where n_x represents the total dimension of the input space F_x. By doing so, each variable of the modeled function can be associated to a specific kernel parameterization as well as a set of specific associated hyperparameters, thus providing more a flexible modeling of the considered function.

In this thesis, the same logic is relied on in order to deal with mixed-variable problems. Rather than defining a kernel on the mixed continuous/discrete design space valid by construction, the developed approach consists in defining two distinct and independent kernels: $k_x(\cdot)$ with respect to the continuous variables $\mathbf{x}$ and $k_z(\cdot)$ with respect to the discrete variables $\mathbf{z}$. Subsequently, the mixed variable kernel $k(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\})$ can be defined as:

$$k(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) = k_x(\mathbf{x}, \mathbf{x}') \times k_z(\mathbf{z}, \mathbf{z}') \tag{3.35}$$

Moreover, the kernel defined above can be further decomposed as a product of one-dimensional kernels, similarly to what is shown in Eq. 3.34. The resulting mixed-variable kernel can then be defined as:

$$k(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) = \prod_{d=1}^{n_x} k_{x_d}(x_d, x_d') \prod_{d=1}^{n_z} k_{z_d}(z_d, z_d') \tag{3.36}$$

Eq. 3.26 describes the construction of a kernel characterizing the covariance between continuous data samples. However, it can be noticed that no assumption on the nature of the considered design variables is required for the kernel to be valid [115]. The kernel construction procedure discussed above can therefore be extended to any type of design variable, as long as a proper mapping and associated Hilbert space can be defined. Examples of kernels defined over non-continuous search spaces can for instance be found within the framework of strings and DNA analysis [75]. In the following section, the construction of kernels defined with respect to discrete variables (regardless of whether qualitative or quantitative) is discussed. As already mentioned in Chapter 2, discrete variables are characterized as non-relaxable variables defined within a finite set of choices, or levels, and characterized by an absence of relation of order between the possible choices.

3.5 Discrete kernels

In order to define kernels allowing to characterize the covariance between the values of a discrete variable $\mathbf{z}$, the basic principle is the same as for continuous variables [119]. The kernel is computed as the inner product between the mappings of the 2 considered discrete variable samples onto a Hilbert space:

$$k_z(\mathbf{z}^i, \mathbf{z}^j) := \langle \phi(\mathbf{z}^i), \phi(\mathbf{z}^j) \rangle_{\mathcal{H}} \tag{3.37}$$

The main difference with what is described in the previous section lies within the fact that the discrete variables that are considered within the framework of system design usually present a finite number of possible values (*i.e.,* levels) and by the fact that the numerical representation assigned to the considered variables levels is usually arbitrary, and does therefore not yield useful information for the kernel construction. By consequence, the mapping of a discrete variable onto a Hilbert space may be different than for a continuous variable as it cannot directly depend on the exact values assigned to the considered variable levels. The discrete kernels discussed in the following paragraphs are defined for one-dimensional inputs (*i.e.,* one dimensional discrete variables). The global discrete kernel can then be computed as the product between the one-dimensional kernels, as already shown in Eq. 3.36:

$$k(\mathbf{z}, \mathbf{z}') = \prod_{d=1}^{n_z} k_{z_d}(z_d, z'_d) \tag{3.38}$$

An alternative way of describing a discrete kernel can be obtained by exploiting the fact that each discrete variable is characterized by a finite number of levels, as is discussed in [137]. By consequence, the kernel function also returns a finite number of covariance values. It is therefore possible to define a $l \times l$ matrix $\mathbf{T}$ containing the covariance values provided by the kernel:

$$\mathbf{T}_{m,d} = k_z(z^i = z_m, z^j = z_d) \tag{3.39}$$

By relying on this approach, the validity of a given kernel $k_z(\cdot)$ can be ensured *a posteriori* by showing that the matrix $\mathbf{T}$ is always symmetric and positive semi-definite for bounded hyperparameter values.

In the following paragraphs, existing kernels for discrete variables are presented. Furthermore, a valid construction under the kernel Hilbert space formalism described by Eq. 3.26 is proposed for each kernel, thus allowing to better highlight and compare their inherent characteristics.

3.5.1 Compound Symmetry

The first and most simple discrete kernel to be considered in this chapter is the Compound Symmetry (CS), characterized by a single covariance value for any non-identical pair of inputs [108]:

$$k(z, z') = \begin{cases} \sigma_z^2 & \text{if} \quad z = z' \\ \theta \cdot \sigma_z^2 & \text{if} \quad z \neq z' \end{cases} \tag{3.40}$$

where σ_z^2 and $0 < \theta < 1$ are respectively the variance and the hyperparameter associated to the CS kernel. By definition, in case the input data samples are identical, the kernel returns the associated variance value σ_z^2, as can be seen in Eq. 3.40. Alternatively, the variance computed between any pair of non-identical data samples is independent from the inputs, and is equal to a value ranging from 0 to σ_z^2. In order to show that the CS is a valid kernel, it is first necessary to

perform the same task with a delta function $\delta(z^i, z^j)$ defined as:

$$k(z, z') = \delta(z, z') = \begin{cases} 1 & \text{if} \quad z = z' \\ 0 & \text{if} \quad z \neq z' \end{cases} \tag{3.41}$$

Let z be a discrete variable characterized by l levels and let $\phi(\cdot)$ be a mapping of the discrete input space onto a l-dimensional Hilbert space: $\phi(z) : F_z \rightarrow \mathbb{R}^l$. The mapping is defined in such a way that the only non-zero coordinate of the image in the Hilbert space corresponds to the dimension associated to the mapped level. This is sometimes referred to as the dummy coding or one-hot coding of a discrete variable [123]. An example of the mapping described above for a generic discrete variable characterized by 4 levels is provided below:

$$z \in \{z_1, z_2, z_3, z_4\} \rightarrow \begin{cases} \phi(z = z_1) = [1, 0, 0, 0] \\ \phi(z = z_2) = [0, 1, 0, 0] \\ \phi(z = z_3) = [0, 0, 1, 0] \\ \phi(z = z_4) = [0, 0, 0, 1] \end{cases}$$

By defining the inner product on the Hilbert space presented above as a standard Euclidean scalar product, the resulting kernel, valid by construction, is equal to the delta function:

$$\langle \phi(z), \phi(z') \rangle = \phi(z)^T \phi(z') = \delta_z(z, z') \tag{3.42}$$

Finally, the CS kernel defined in Eq. 3.40 can be defined through the following combination of delta function and constant kernels:

$$k(z, z') = \sigma_z^2 \left(c_2 k_1(z, z') + c_1 \right) = \sigma_z^2 \left((1 - \theta) \delta_z(z, z') + \theta \right) \tag{3.43}$$

The kernel above is always valid as long as the constants c_1 and c_2 are positive, which is ensured by bounding $\theta \in [0, 1]$. It should be pointed out that the CS kernel presented in the previous paragraphs is nearly identical to the one proposed by Hutter and Halstrup in [54] and [62], although the construction differs. The approach proposed in these works consists in defining a distance in the mixed-variable search space by relying on the concept of Gower distance [51]:

$$d_{gow}\left(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\} \right) = \frac{\sum_{d=1}^{n_x} \frac{|x_d - x'_d|}{\Delta x_d}}{n_x} + \frac{\sum_{d=1}^{n_z} \delta(z_d, z'_d)}{n_z} \tag{3.44}$$

where Δx_d is the domain range of the continuous variable x_d. This distance can then be rescaled and used in order to compute the covariance as a function of the Gower distance between data samples by relying on a continuous squared exponential kernel (or any other distance based kernel):

$$k(z, z') = \sigma_z \exp\left(-\theta d_{gow}\left(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\} \right)^2 \right) \tag{3.45}$$

It can be easily shown that by selecting the appropriate parameterization, the results provided by the kernels defined in Eq. 3.40 and 3.45 are identical.

The CS kernel presented in the paragraph above provides a very simple method allowing to model the effect of a given discrete design variable by relying on a single hyperparameter. However, the underlying assumption which is made when considering the CS kernel is that the covariance between any pair of non identical levels of a given discrete variable is the same, regardless of the considered levels. This assumption may often be overly simplistic, especially when dealing with discrete variables which present a large number of levels. In this case, the modeling error introduced by CS kernel can become problematic, and alternative kernels should be considered. It

is also important to point out that the CS kernel formulation, as presented in Eq. 3.40, can only return positive covariance values (by construction). This characteristic further limits the number of suitable applications for this particular covariance function. Finally, it is worth mentioning that Roustant *et al.* have extended the CS kernel in order to model mixed-variable functions characterized by discrete variables with a large number of levels [108]. The underlying idea is to group levels with similar characteristics, thus allowing to compute the covariance between these groups rather than between the levels. However, given that in the scope of this work the considered problems present a limited number of levels, this approach is not further discussed.

3.5.2 Hypersphere decomposition kernel

A second discrete kernel considered in this work is the hypersphere decomposition kernel, first proposed by Zhou *et al.* [137]. The working principle of the kernel is based on mapping each of the l levels of the considered discrete variable onto a distinct point on the surface of a l-dimensional hypersphere:

$$\begin{aligned} &\phi(z) : F_z \rightarrow \mathbb{R}^l \\ &\phi(z = z_m) = \sigma_z [b_{m,0}, b_{m,1}, \dots, b_{m,l}]^T \quad \text{for } m = 1, \dots, l \end{aligned} \tag{3.46}$$

where $b_{m,d}$ represents the d-th coordinate of the m-th discrete variable level mapping, and is computed as follows:

$$\begin{aligned} b_{m,d} &= 1 && \text{for } m \text{ and } d = 1 \\ b_{m,d} &= \cos\theta_{m,d} \prod_{k=1}^{d-1} \sin\theta_{m,k} && \text{for } d = 1, \dots, m-1 \\ b_{m,d} &= \prod_{k=1}^{d-1} \sin\theta_{m,k} && \text{for } d = m \neq 1 \\ b_{m,d} &= 0 && \text{for } d \geq m \neq 1 \end{aligned}$$

with $-\pi \leq \theta_{m,d} \leq \pi$. It can be noticed that in the equations above, some of the mapping coordinates are arbitrarily set to 0. This allows to avoid rotation indeterminacies (*i.e.*, an infinite number of hyperparameter sets characterizing the same covariance matrix), while also reducing the number of parameters required to define the mapping. The resulting kernel is then computed as the Euclidean scalar product between the hypersphere mappings presented above:

$$k(z, z') = \phi(z)^T \phi(z') \tag{3.47}$$

This kernel construction defined above is equivalent to the original formulation [137], in which the discrete kernel is defined as an $l \times l$ symmetric positive definite matrix $\mathbf{T}$ containing the covariance values between the discrete variable levels. In order to ensure the positive definiteness of this matrix, it is defined through the following Cholesky decomposition:

$$\mathbf{T} = \mathbf{L}^T \mathbf{L} \tag{3.48}$$

where each element of $\mathbf{L}_{i,j}$ is computed as $b_{i,j}$:

$$\mathbf{L} = \sigma_z \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ \cos\theta_{2,1} & \sin\theta_{2,1} & 0 & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cos\theta_{l,1} & \sin\theta_{l,1}\cos\theta_{l,2} & \dots & \cos\theta_{l,l-1}\prod_{d=1}^{l-2}\sin\theta_{l,d} & \prod_{d=1}^{l-1}\sin\theta_{l,d} \end{bmatrix} \tag{3.49}$$

Differently than the CS kernel, the hypersphere decomposition kernel can return a different covariance value for each pair of levels characterizing the considered discrete variable. Furthermore, with the proper hyperparameters this kernel function can also return negative values, as each covariance value is computed as the product between a number of sine and cosine functions, which range from -1 to 1. However, it should also be highlighted that a part of the hyperparameters characterizing this kernel (*i.e.,* $\theta_{l,d}$) influence several covariance values simultaneously and that furthermore, part of the covariance values can depend on several hyperparameters simultaneously. As a result, determining the optimal value of each hyperparameter might be more complex when compared to simpler kernel parameterizations such as CS.

3.5.3 Latent variable kernel

Similarly to the hypersphere decomposition kernel, the Latent Variable (LV) kernel, first proposed by Zhang *et al.* [135], is constructed by mapping the discrete variable levels onto a continuous Hilbert space. However, in the latent variable approach, the discrete variable levels are mapped onto a 2-dimensional Euclidean space regardless of the number of discrete levels, rather than on the surface of an l-dimensional hypersphere. This mapping can be defined as follows:

$$\begin{aligned} &\phi(z) : F_z \rightarrow \mathbb{R}^2 \\ &\phi(z = z_m) = [\theta_{m,1}, \theta_{m,2}]^T \quad \text{for } m = 1, \dots, l \end{aligned} \tag{3.50}$$

where $\theta_{m,1}$ and $\theta_{m,2}$ are the hyperparameters representing coordinates in the 2-dimensional latent variable space onto which the discrete variable level m is mapped. By consequence, the set of hyperparameters characterizing this kernel is represented by the l pairs of latent variable coordinates associated to a given discrete variable. For clarity purposes, an example of the mapping of a discrete variable characterizing the material choice property of a hypothetical system is provided in Figure 3.5.

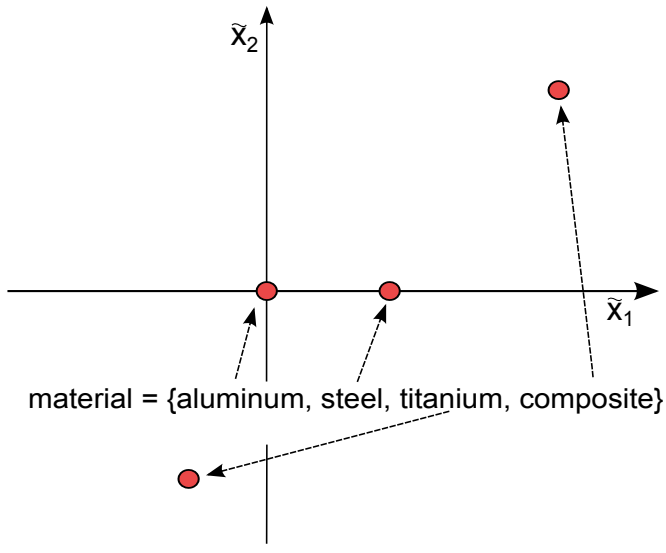


FIGURE 3.5: Example of latent variable mapping for a generic discrete variable characterizing the 'material choice' characteristic.

Let $\phi : F_z \rightarrow \mathbb{R}^2$ be the mapping defined in Eq. 3.50, by applying the mapping rule discussed in Section 3.4.1, it can be shown that a kernel $\tilde{k}(\cdot)$ valid on $\mathbb{R}^2$ can be used in order to define a

kernel $k(\cdot)$ valid on F_z in the following fashion [119]:

$$k(z, z') = \tilde{k}(\phi(z), \phi(z')) \tag{3.51}$$

By consequence, any of the continuous kernels valid on $\mathbb{R}^2$ can be coupled with the latent variable mapping in order to define a valid kernel on the discrete search space F_z. Although in the original formulation of the method, the following squared exponential kernel is considered:

$$k(z, z') = \sigma_z^2 \exp(||\phi(z) - \phi(z')||_2^2) \tag{3.52}$$

alternative continuous kernels could technically be used without loss of generality. It can be noticed that no lengthscale parameter θ appears in the squared exponential kernel as defined in Eq. 3.52. The reason behind this is that the distance between the latent variables already directly depends on the hyperparameter values (*i.e.,* the latent variables coordinates), and by consequence a lengthscale parameter would be redundant. Similarly to the hypersphere decomposition kernel previously discussed, the latent variable kernel requires removing the translation and rotation indeterminacies on the latent variable value estimations. Zhang *et al.* suggest fixing one of the latent variables coordinate pair to the origin of the 2-dimensional latent search space (*e.g.,* $\{\theta_{0,1}, \theta_{0,2}\} = \{0, 0\}$), and a second pair on the θ_1 axis (*i.e.,* $\theta_{1,1} = 0$).

The latent variable kernel construction discussed above relies on mapping the discrete variable levels onto a 2-dimensional Hilbert space. This choice is arbitrary as the mapping can technically be performed onto a higher dimensional space in order to provide a larger flexibility and improve the modeling accuracy of the model. However, Zhang *et al.* [135] state that the theoretical improvement of modeling performance does not compensate the increase in the number of hyperparameters to be tuned, and identify therefore the 2-dimensional latent space as the optimal trade-off between the kernel modeling capabilities and the number of associated hyperparameters. Similarly to the hypersphere decomposition kernel, the latent variable kernel allows to define a distinct covariance value between every pair of levels. However, due to the fact the covariance values are computed as a function of the distance between latent variables in a continuous space, as shown in Eq. 3.52, the returned values can not be negative, which partially limits the modeling capabilities of the LV discrete kernel.

By mapping the levels of the considered discrete variable onto a 2-dimensional latent space and by characterizing the covariance between these levels as the Euclidean distance between the latent variables, the latent variable kernel provides an intuitive visual representation of how the levels are correlated to each other, as is shown in Figure 3.5. In this figure, the distance between the latent variables (*i.e.,* red dots) associated to the various materials is inversely proportional to the covariance between the relative levels. For instance, with the considered hyperparameter values, the computed covariance between the aluminum and steel choices would be larger than the one between the aluminum and the composite choices.

3.5.4 Coregionalization

The last discrete kernel considered in this work is based on the definition of a coregionalization matrix [6]. This approach is originally developed for the modeling of vector valued functions with respect to a continuous search space, *i.e.,* functions returning vector outputs rather than scalar values, $\mathbf{f} : F_x \rightarrow \mathbb{R}^{n_{outputs}}$, where $n_{outputs}$ is the size of the output vector. The underlying idea of the original formulation is to exploit the existing correlation between the various outputs in order to improve the modeling accuracy with respect to the separate and independent modeling of each output. For purely continuous functions, the Linear Model of Coregionalization (LMC) approach, as defined in [64], consists in computing each component prior f_d of the prediction vector as a sum of Q groups of independent latent GP models $u_q^i(\mathbf{x})$, where each GP group q of

size R_q shares the same covariance function:

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{Rq} a_{d,q}^i u_q^i(\mathbf{x}) \tag{3.53}$$

with $a_{d,q}^i$ being scalar coefficients. When considering this kind of models, the resulting vector valued kernel (*i.e.,* one covariance function value per output) can be written as:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} \mathbf{B}_q k_q(\mathbf{x}, \mathbf{x}') \tag{3.54}$$

where $k_q(\cdot)$ is the kernel associated to the group of GP u_q, while $\mathbf{B}_q$ is the coregionalization matrix, containing the elements $b_{d,d'}^q$ which characterize the cross-covariance between the predictions d and d' associated to u_q. The LMC can be simplified into the Intrinsic Coregionalization Model (ICM) by considering a single kernel (*i.e.,* $Q = 1$) [50], which results in the following expression of the multiple output covariance matrix:

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes \mathbf{k}(\mathbf{X}, \mathbf{X}) \tag{3.55}$$

where $\otimes$ is the Kronecker product between matrices and $\mathbf{k}(\mathbf{X}, \mathbf{X})$ is the Gram covariance matrix computed on the continuous part of the training data set. The equation above is equivalent to:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \mathbf{B} \times k(\mathbf{x}, \mathbf{x}') \tag{3.56}$$

Without loss of generality, the concept of coreginalization can be applied to the modeling of mixed variable problems by considering the modeled function as returning an independent output for each level of the considered discrete variable. In this case, the coregionalization matrix becomes the matrix containing the covariance values between the discrete variable levels.

$$\mathbf{B}_{m,d} = k_z(z = z_m, z' = z_d) = \mathbf{T}_{m,d} \tag{3.57}$$

In order for the ICM to represent a valid kernel, it is necessary for the coregionalization matrix $\mathbf{B}$ to be symmetric and positive semi-definite. Similarly to the hypersphere decomposition, the kernel defined above can be obtained by mapping the levels of the considered discrete variable onto an l-dimensional continuous space:

$$\begin{aligned} &\phi : F_z \rightarrow \mathbb{R}^l \\ &\phi(z = z_m) = [\theta_{m,0}, \theta_{m,1}, \dots, \theta_{m,l}]^T \quad \text{for } m = 1, \dots, l \end{aligned} \tag{3.58}$$

However, differently than for the hypersphere decomposition kernel, the Cartesian coordinates of the locations onto which the discrete levels are mapped are directly considered as hyperparameters. By defining the inner product on the l-dimensional Hilbert space a Euclidean scalar product, a valid discrete kernel can then be defined:

$$k(z, z') = \phi(z)^T \phi(z') \tag{3.59}$$

It can be shown that Eq 3.55 and Eq. 3.59 yield the same covariance values by exploiting the fact that any positive semi-definite matrix $\mathbf{B}$ can be obtained as the product between a real valued matrix $\mathbf{W}$ and its transpose [101]:

$$\mathbf{B} = \mathbf{W}^T \mathbf{W} \tag{3.60}$$

If $\mathbf{W}$ is defined as the $l \times l$ matrix containing the hyperparameters onto which each level of the considered discrete variable is mapped, the equivalence between Eq. 3.55 and Eq. 3.59 within the scope of coregionalization applied to discrete variable kernels is shown.

It is important to point out that when adapting the ICM to the modeling of mixed-variable functions, the underlying assumption is that only one discrete variable is considered, the levels of which are associated to independent outputs of a vector valued function. However, the kernel defined in the paragraphs above is, by construction, valid on a discrete dimension, regardless of the presence of other discrete variables. It can therefore be applied in a multi-dimensional discrete search space framework without loss of generality. Similarly to the hypersphere decomposition case, the coregionalization approach allows to characterize both positive and negative covariance values between the discrete variable levels. Finally, it can be noticed that differently than the hypersphere decomposition and latent variable kernels, the coregionalization kernel provides by construction an independent variance value for each level of the modeled variable. This resulting property, known as heteroscedasticity, is further discussed in Section 3.6.2.

3.5.5 Discrete kernel comparison

The various discrete kernels described in the previous paragraphs present specific advantages and weaknesses which must be taken into account when selecting the most suitable kernel parameterization for the modeling of a given problem. For clarity purposes, the main characteristics of the presented discrete kernels are summarized in Table 3.1. It can be noticed that the scaling of

Kernels	Hyperparameter scaling w.r.t. l	Different covariance per level pair	Negative covariance values	Inherently heteroscedastic
CS	1	No	No	No
HS	$l(l-1)/2$	Yes	Yes	No
LV	$2l-3$	Yes	No	No
CN	l^2	Yes	Yes	Yes

TABLE 3.1: Advantages and weaknesses of discrete kernel parameterizations. The acronyms refer to the following kernels: CS: Compound Symmetry, HS: Hypersphere Decomposition, LV: Latent Variable, CN: Coregionalization.

the number of hyperparameters with respect to the number of levels of the considered discrete variable varies considerably between the various kernels. This must be taken into account when selecting the appropriate kernel, as a large set of hyperparameters might be difficult to tune when relying on an insufficient amount of training data. On the contrary, simplistic kernel parameterizations, such as the CS, might be inadequate for variables with large number of levels in case sufficient data is available. It can also be noticed that the coregionalization kernel is the only one which allows to provide a heteroscedastic GP model. However, this limitation can be partially solved by including an additional term in the considered kernel which results in a different and independent variance associated to each discrete level. The description of this extension can be found in Section 3.6.2.

3.6 Considerations on mixed-variable Gaussian Processes

3.6.1 Category-wise and level-wise mixed-variable kernels

In the previous paragraphs, only kernels for one-dimensional discrete vectors are discussed. However, most of the considered problems depend on multiple discrete variables, in which case two different approaches can be chosen: treating each dimension (*i.e.,* each discrete variable)

separately and independently or defining a kernel on the combinatorial discrete search space. In this thesis, these approaches are referred to as Level-wise and Category-wise kernels.

The Level-wise approach is based on considering each discrete variable separately and defining an independent kernel for each one of them. In this case, the various kernels are tasked with computing the covariance between the various levels of each variable. This allows to more easily identify the specific influence of the considered discrete variables and thus select the most suitable kernels, although this might require some previous knowledge on the modeled function. Furthermore, the resulting model usually requires fewer hyperparameters in order to be defined. However, dealing with each dimension separately implies considering that each dimension is independent from the others [104]. In practice, this means that the covariance between two levels (*i.e.,* possible values) of a given discrete variable is independent from the similarity of the 2 compared samples with respect to the other discrete variables. In some cases, this assumption may be false. For instance, if the propulsion sub-system of a launch vehicle is considered, discrete variables such as the type of propulsion (*i.e.,* solid, liquid) and the number of engines influence the performance of the system in a joint fashion, rather than independently.

Alternatively, the Category-wise kernel definition allows to avoid this kind of issues. The main idea is to define the kernel on the combinatorial discrete search space rather than separately on each dimension of the input space. By doing so, the resulting kernel allows to compute the covariance between discrete variable levels combinations (*i.e.,* categories) rather than between discrete variable levels. However, the drawback of this approach is that the number of hyperparameters required to describe such a kernel tends to scale exponentially with the number of discrete variables. By consequence, larger amounts of data are usually required in order to properly learn the modeled function trends. If this condition is not satisfied, this solution tends to provide worse modeling performance than the level-wise approach. A second drawback of the category-wise approach is that data samples belonging to every category of the considered problem are necessary in order to train the model. This can be problematic when dealing with computationally intensive problems characterized by a large number of categories. Instead, the level-wise approach allows to extrapolate the missing data from other samples when considering categories which are not present within the data set. Finally, it can be noted that when considering a category-wise approach, the input of the global discrete kernel can be represented under the form of a scalar (*i.e.,* a single discrete variable) characterizing the category the considered sample belongs to rather than the level values of its discrete variables.

It is extremely important to keep in mind that category-wise modeling and category-wise kernel are two distinct and different approaches for the modeling of mixed-variable functions. Indeed, category-wise modeling refers to the creation of a separate and independent continuous GP for every category of the considered function, whereas the category-wise kernel approach refers to the creation of a discrete kernel allowing to compute the covariance between the categories of the considered discrete variables.

3.6.2 Surrogate model scedasticity

Within the framework of continuous GP, it is common practice to consider the model as being homoscedatic, which translates to a constant variance value with respect to the design space:

$$k(\mathbf{x}^*, \mathbf{x}^*) = \sigma^2 \tag{3.61}$$

When the GP kernel is defined as the product of one-dimensional continuous and discrete kernels, as in Eq. 3.36, the homoscedastic variance can be interpreted as the product between the variances associated to each one of the kernels:

$$\sigma^2 = \prod_d^{n_x} \sigma_x^2 \prod_d^{n_z} \sigma_z^2 \tag{3.62}$$

Although the homoscedasticity assumption is often acceptable in the purely continuous case, it may be overly simplistic in the mixed continuous/discrete case, as the discrete variables may be associated to large variations of the modeled function global trend. In these cases, it may be necessary to model the kernel variance so that its value depends on the input data sample, thus obtaining what is referred to as a heteroscedastic GP. In the most general case, the heteroscedastic mixed-variable GP variance should vary as a function of both continuous and discrete variables. However, due to the fact that the thesis focus is mainly on the discrete aspects of mixed-variable GP and in order to reduce the model training complexity, in this work the variance is only considered to vary as a function of the discrete variables $\mathbf{z}$:

$$k(\{\mathbf{x}^*,\mathbf{z}^*\},\{\mathbf{x}^*,\mathbf{z}^*\}) = \sigma^2(\mathbf{z}^*,\mathbf{z}^*) \tag{3.63}$$

In general, a prior knowledge of the global trend of the modeled function allows to determine which choice, between a homoscedastic and a heteroscedastic model is more suitable for a given problem. In this chapter, it is assumed that no prior information on the modeled function is known, and both alternatives are tested for the considered kernel parameterizations. Finally, it is important to note that considering a heteroscedastic discrete kernel has a different influence for Level-Wise and Category-Wise kernels, due to the fact that in the first case a different variance value is associated each level of each discrete variable, whereas in the latter a different variance value is associated to each category of the considered problem.

The heteroscedasticity of a mixed-variable function can be easily shown by considering the simple test-function shown in Figure 3.6 and defined as follows:

$$f(x,z) = \begin{cases} \sin(7x) & \text{if} \quad z = 0 \\ 2\sin(7x) & \text{if} \quad z = 1 \end{cases} \tag{3.64}$$

The variances associated to the 2 categories of the considered function (which in this case are

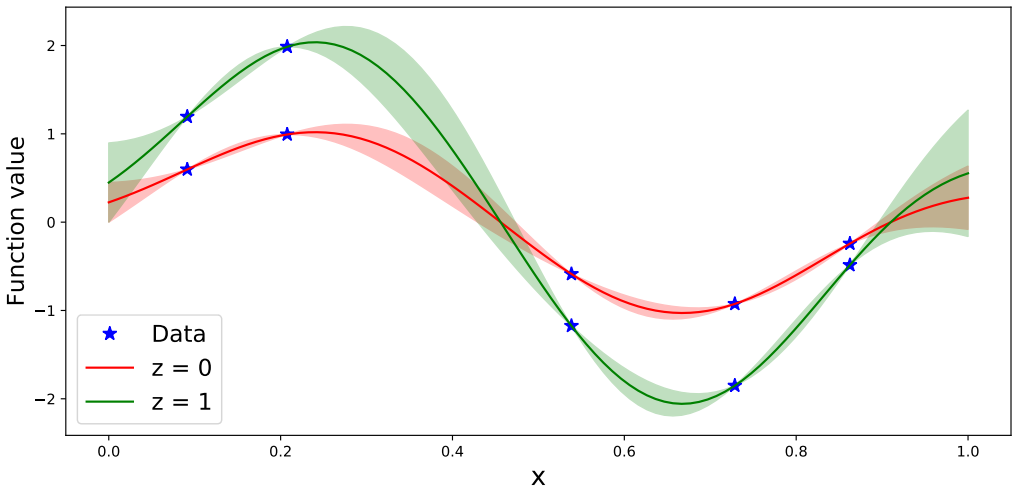


FIGURE 3.6: Category-wise independent modeling of a heteroscedastic mixed variable function

equivalent to the levels of the variable z) can be compared by modeling it with 2 independent GP models obtained by considering identical continuous data sets, as shown in Figure 3.6. In general, the optimal variance value depends on the size and on the values of the specific data set with respect to which the models are trained. However, by repeating the comparison over several data sets it can be shown that on average the variance associated to the first category of the problem (*i.e.*, $z = 0$) is approximately 4 times smaller than the one associated to the second category of

the problem (*i.e.*, $z = 1$). This is highlighted in Figure 3.7, where the estimated optimal variance associated to the 2 categories computed over 20 repetitions is provided for 3 different data set sizes. The results show that in general, a heteroscedastic mixed-variable kernel should therefore

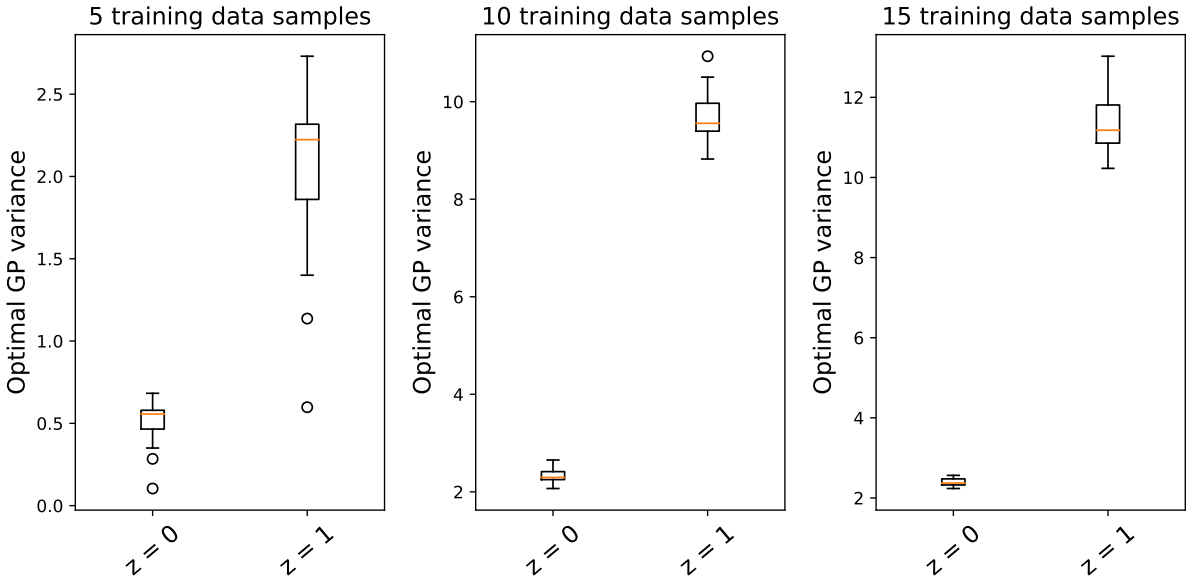


FIGURE 3.7: Estimated optimal variance associated to the two categories of a heteroscedastic example for various data set sizes over 20 repetitions per size

yield more accurate predictions when dealing with functions presenting similar characteristics.

The heteroscedastic variance function defined in Eq. 3.63 can be characterized as a discrete kernel, similar to the ones presented in the previous paragraphs:

$$\sigma^2(\mathbf{z}^i, \mathbf{z}^j) = \prod_{d=1}^{n_z} k_{z_d}(z_d^i, z_d^j) \tag{3.65}$$

where each kernel k_{z_d} can be constructed by mapping each level characterizing the discrete variable z_d onto a hyperparameter characterizing the variance associated to the level:

$$\begin{aligned} &\phi(z) : F_z \rightarrow \mathcal{R} \\ &\phi(z_m) = \theta_m \quad \text{for } m = 1, \dots, l \end{aligned} \tag{3.66}$$

and by defining the inner product as a standard product between scalars:

$$k_(z^i, z^j) = \langle \phi(z^i), \phi(z^j) \rangle = \phi(z^i)\phi(z^j) \tag{3.67}$$

The resulting variance function provides a better flexibility and subsequent modeling performance when dealing with functions characterized by heteroscedastic behaviors (with respect to the discrete design variables). However, considering a heteroscedastic discrete kernel also results in a larger number of hyperparameters to be trained. Therefore, relying on a heteroscedastic kernel in order to model a homoscedastic function when insufficient data is provided might be counterproductive and yield worse results than a homoscedastic model.

3.6.3 Noisy data modeling

In some particular applications, the training data used to create the GP may be affected by noises of various nature. In general, the commonly encountered noisy functions can be represented as a sum between a deterministic term $f(\cdot)$ and a noise term η [107]:

$$y = f(\mathbf{x}, \mathbf{z}) + \eta \tag{3.68}$$

The noise η can have different causes, such as truncation errors, measuring errors and simulation codes with non-converging internal optimization process. Relying on such data in order to train a GP may cause the model to over-fit said data, thus resulting in a possibly large modeling error. In order to take into account the possible presence of noisy data, it is common practice to add a noise handling term to the global kernel:

$$Cov\left(Y(\mathbf{x}, \mathbf{z}), Y(\mathbf{x}', \mathbf{z}')\right) = \prod_{d=1}^{n_x} k_{x_d}(x_d, x_d') \prod_{d=1}^{n_z} k_{z_d}(z_d, z_d') + \sigma_n^2 \delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) \tag{3.69}$$

where σ_n^2 is a noise handling hyperparameter (usually proportional to the noise magnitude) and δ_n is a kernel function similar to the Kronecker delta defined as:

$$\delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) = \begin{cases} 1 & \text{if } \{\mathbf{x}, \mathbf{z}\} = \{\mathbf{x}', \mathbf{z}'\} \\ 0 & \text{if } \{\mathbf{x}, \mathbf{z}\} \neq \{\mathbf{x}', \mathbf{z}'\} \end{cases} \tag{3.70}$$

Depending on the considered research domain, the noise handling term is often referred to as nugget or likelihood variance [107]. By properly tuning the value of the hyperparameter σ_n^2 it is possible to improve the robustness of the considered GP model with respect to noisy training data and thus avoid over-fitting issues [107]. However, it is important to highlight the fact that inclusion of a nugget in the kernel also results in a non-interpolating surrogate model. Alternatively, the same results can also be obtained by adding an $n \times n$ identity matrix $\sigma_n^2 \mathbf{I}$ to the Gram covariance matrix $\mathbf{K}$. Finally, please note that the derivation of the validity of $\delta_n(\cdot)$ can be found in Appendix B.

3.6.4 Hyperparameter optimization

The majority of the continuous and discrete kernels presented in the previous paragraphs depends on a number of hyperparameters, such as the lengthscales of Eq. 3.28 and the latent variables coordinates of Eq. 3.52, which influence the value returned by the kernel function, independently from the input data samples. Furthermore, the global kernel function also depends on additional parameters: namely the kernel and likelihood variances (σ^2, σ_n^2) as well as the GP mean μ. Although in some approaches the global kernel variance and the mean are expressed as a function of the training data and the correlation matrix [63], in this work all of these 3 parameters are optimized independently in the same way as the continuous and discrete kernel hyperparameters.

In order for the GP to provide the most accurate prediction of the modeled function and a valid associated error model, the optimal value of every hyperparameter mentioned in the paragraph above must be determined. This process is often referred to as the GP training. Several methods and criteria exist for the training of a GP, such as the cross-validation [26], the marginal likelihood optimization [107] and the restricted likelihood optimization [30]. In this thesis, the marginal likelihood optimization is used. Let $\boldsymbol{\theta}$ be the vector containing all the hyperparameters characterizing the global kernel $k(\cdot)$, the contained values are determined by

maximizing the log marginal likelihood:

$$\begin{aligned} \boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\log p(\mathcal{Y}|\mathbf{X},\mathbf{Z},\boldsymbol{\theta})\right) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-\frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log 2\pi\right) \end{aligned} \tag{3.71}$$

where $\mathbf{K}$ is computed by relying on kernels which are parameterized as a function of $\boldsymbol{\theta}$. Although the hyperparameter vector $\boldsymbol{\theta}$ characterizes both continuous and discrete kernels, all of its values are defined within a continuous search space, thus allowing to rely on common continuous optimization methods. For all the results presented in this work, the optimization problem defined in Eq. 3.71 is solved with the help of a Bounded Limited memory Broyden - Fletcher - Goldfarb – Shanno (L-BFGS-B) algorithm [24], which is a memory-efficient variant of the BFGS quasi-Newton optimization method [39]. Similarly to other gradient-based optimization algorithm, the L-BFGS-B can usually provide a fast convergence towards the log-likelihood optimization problem optimum. However, like most optimization methods of this family, the convergence towards the global optimum of the problem is dependent on the initialization in the design space and cannot always be ensured. This issue is particularly relevant when considering the discrete kernel parameterizations discussed in the previous sections. Generally speaking, in order to maximize the chances for the likelihood optimization to converge towards the global optimum it is necessary to initialize the hyperparameters characterizing the covariances between discrete variable levels as close to the optimal solution as possible. However, unless some specific knowledge regarding the considered modeled function is known, this is usually not possible. For illustrative purposes, the material choice mapping introduced in Figure 3.5 is considered anew. Let the hypothetical initialization of the latent variable coordinate hyperparameters and the optimal solution be presented in the left and right graphics of Figure 3.8, respectively. It is fair to assume that the likelihood landscape might present several local optima, especially in the part of the search space where the latent variables associated to the titanium and steel choices are close one another, which the gradient-based algorithm might very likely cross with the given initialization (left side). Similar issues also arise when all the hyperparameters are initialized at the same value, which in the latent variable case corresponds to identical covariance values between all the levels. In general,

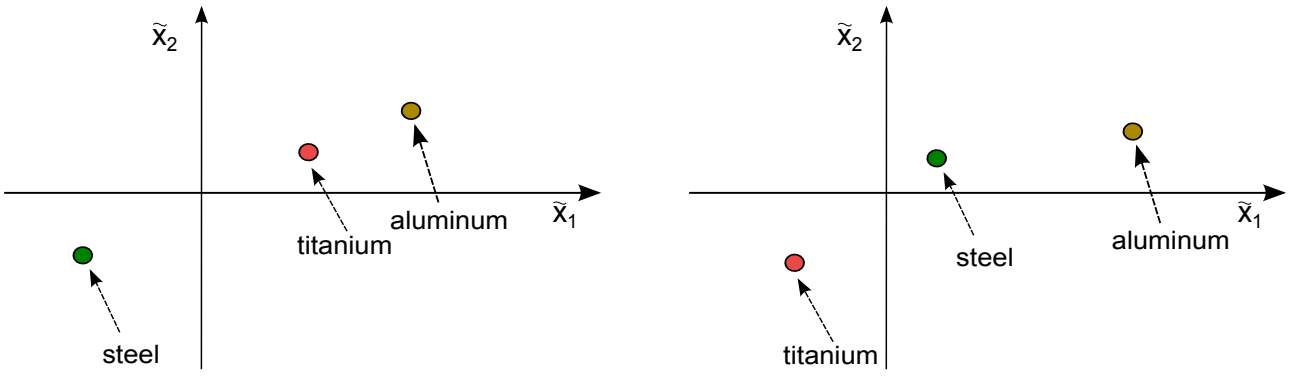


FIGURE 3.8: Hypothetical initialization (left) and optimal solution (right) for latent variables hyperparameters characterizing the covariance between material choices.

depending on the problem at hand as well as on the considered kernel parameterization, this issue might need to be dealt with. From experience, a random uniform initialization of the discrete kernel hyperparameters (within their bounds) is sufficient in order to ensure a reasonably robust convergence of the likelihood optimization. However, in some particular cases a multiple random

initialization of the algorithm with the selection of the result characterized by the largest likelihood value might be necessary. In order to avoid this issue altogether, heuristic optimization methods, such as the Differential Evolution [122] or the Covariance Matrix Adaptation Evolution Strategy [55], were also considered and tested. However this option was not viable due to the prohibitive increase of the overhead computational cost related to the GP training when dealing with large Gram covariance matrices (usually associated to large dimensional problems).

3.6.5 Mixed-variable design of experiments

The modeling accuracy which can be provided by a GP is closely related to the training data set given as an input to the model, which is sometimes referred to as Design of Experiments (DoE). In general, larger data sets result in a better modeling accuracy. However, given that this thesis lies within the scope of complex and computationally intensive system design, the amount of data available for the creation of the surrogate models is usually very limited. It might therefore be necessary to distribute the available data over the design space in such a way to maximize the information that the GP model can use. In the purely continuous case, the data samples can either be sampled on given distributions, such as a uniform distribution, or alternatively slightly more advanced sampling methods, such as the Latin Hypercube Sampling (LHS) [86] method can be used. In order to extend the concept of design of experiments in the mixed continuous/discrete search space, the assumption that is made in this thesis is that in the absence of problem specific-knowledge, the same amount of data should be placed in each category characterizing the modeled function. Under this assumption, 3 possible alternatives for the creation were considered:

- Evenly and randomly distributing the data obtained through a single continuous sampling between all of the problem categories.
- Replicating the same continuous sampling in each category of the considered problem.
- Performing an independent continuous sampling in each category of the considered problem.

In general, samples which are close one another in the continuous search space while presenting different discrete variable values allow the GP model to better determine the covariances between the various discrete levels and categories. On the other hand, spacing the samples within the continuous design space allows to better capture the global trend of modeled function in the continuous domain. The first mixed-variable DoE option allows to optimize the usable information within the continuous part of the design space, while limiting the GP ability to determine the covariance between the problem categories. Alternatively, the second option can provide a more accurate computation of the covariance between categories, however it also tends to increase the modeling error for unmapped locations which are distant from the training data set samples, due to the poor coverage of the continuous search space. Finally, the third alternative represents a compromise between the first two options. Further analysis of the challenges related to the mixed-variable sampling issues can be found in [28].

Usually, an analysis of the considered problem and of the amount of available computational resources is required in order to select the most appropriate approach for the creation of a mixed-variable DoE. However, this type of analysis extends beyond the scope of the thesis. In this work, the first approach combined with a continuous LHS method is considered for the DoE creation.

3.7 Modeling performance comparison

In the previous section, different kernels allowing to characterize the covariance function between discrete variables levels are presented, discussed and compared. In order to assess the

modeling performance of the various kernels as well as its dependence on the characteristics of the considered function, a number of analytical and engineering related test-cases characterized by different dimensions, complexities and characteristics are considered. The comparison between the kernels is based on an analysis of the modeling error (*i.e.,* difference between the actual function value and the GP prediction) which is provided under the form of a Root Mean Squared Error (RMSE) calculated on a validation data set distinct from the training one:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}(\mathbf{x}^i,\mathbf{z}^i) - y(\mathbf{x}^i,\mathbf{z}^i)\right)^2} \tag{3.72}$$

where N is the size of the validation data set and the values of $y(\cdot)$ and $\hat{y}(\cdot)$ are normalized between 0 and 1:

$$\hat{y}_{norm} = \frac{\hat{y} - y_{min}}{y_{max} - y_{min}} \tag{3.73}$$

y_{min} and y_{max} represent respectively the smallest and largest values present within the training data set (*i.e.,* in the vector $\mathbf{y}$). In order to analyze the change in absolute and relative performance of the various kernels, the modeling benchmark is performed for different training data set sizes, when possible. Furthermore, in order to take into account the variability of the obtained results caused by the random nature of the training DOE, the benchmark is repeated multiple times for each training data set size (varying as a function of the computational cost and complexity of the problem). The validation data set size is fixed to 1000, and its elements are generated randomly by combining a continuous LHS and a sampling over a uniform discrete distribution in the discrete search space.

In the presented benchmark, the compared discrete kernels are the following: Compound Symmetry (CS), Hypersphere decomposition (HS), Latern Variables (LV) and Coregionalization (CN). Furthermore, for each one of these kernels, 4 alternative variants are considered: The homoscedastic level-wise approach, the heteroscedastic level-wise approach (referred to as _He), the homoscedastic category-wise approach (referred to as _C) and finally the heteroscedastic category-wise approach (referred to as _C_He). However, due to the inherently heteroscedastic nature of the CN kernel, only the level-wise and category-wise variants are considered in the presented results. In order to provide a modeling performance reference, the modeling benchmark is also performed by relying on a Category-Wise (CW) GP, which is obtained by defining an independent purely continuous GP for every category of the considered function. Each one of these GP is trained by relying solely on the data samples of the training set which belong to the corresponding category. The CW GP prediction of the modeled function at an unmapped location is then computed by evaluating the continuous GP corresponding to the sample category at the location characterized by the sample continuous variables. This results in a total number of compared methods equal to 15. Given that the main focus and contribution of this thesis is related to the discrete part of mixed-variable kernel, the continuous kernel considered in order to obtain the results presented in this chapter and throughout this thesis is the squared exponential kernel previously described in Eq. 3.27. Finally, it might be worth mentioning that in the heteroscedastic CS parameterization case, the hyperparameters characterizing the heteroscedastic kernel outnumber and outweigh the CS kernel-specific ones. By consequence, the results obtained with the level-wise and category-wise heteroscedastic CS kernel might be mainly driven by the heteroscedastic aspect rather than by the specific kernel parameterization.

3.7.1 Benchmark analysis

In the following paragraphs, the various discrete kernels presented in this chapter are tested on a number of benchmarks. More specifically, 5 analytical functions and 2 engineering-related

test-cases are considered. These benchmarks present different characteristics in terms of continuous and discrete design space dimensions, combinatorial design space size, complexity, presence of negative correlation and heteroscedastic trends and category-wise construction. The key properties (from a modeling perspective), simulation details and expected analyses for each benchmark are provided below:

Branin function

- 2 continuous dimensions, 2 discrete dimensions, 9 categories
- Modeling performed for 3 data set sizes: 20, 40, 80
- Presence of negative correlation trends between levels and category-wise construction of the function.

Augmented Branin function

- 10 continuous dimensions, 2 discrete dimensions, 4 categories
- Modeling performed for 3 data set sizes: 80, 160, 320
- Increase of the continuous design space size with respect to the Branin function (but identical discrete design space characteristics). Presence of negative correlation trends between levels and category-wise construction of the function.

Goldstein function

- 2 continuous dimensions, 2 discrete dimensions, 9 categories
- Modeling performed for 3 data set sizes: 27, 72, 135
- Increase in the number of categories with respect to the Branin function (but identical discrete design space size). Overall homoscedastic trend.

Analytical benchmark N.4

- 1 continuous dimensions, 2 discrete dimensions, 8 categories
- Modeling performed for 3 data set sizes: 40, 80, 120
- Overall homoscedastic with negative correlation trends between levels.

Analytical benchmark N.5

- 5 continuous dimensions, 5 discrete dimensions, 243 categories
- Modeling performed for 3 data set sizes: 60, 90, 120
- Large number of categories, impossibility of relying on category-wise approaches.

Propulsion performance simulation

- 3 continuous dimensions, 2 discrete dimensions, 7 categories (16 theoretical)
- Modeling performed for 3 data set sizes: 21, 56, 105
- Realistic simulation. Not all level combinations are physically feasible, which results in not all categories being present in the DoE.

Thrust frame structural analysis

- 12 continuous dimensions, 2 discrete dimensions, 9 categories
- Modeling performed for 1 data set sizes: 135
- Realistic simulation. Linear trends with respect to the continuous sizing variables.

Implementation
The results presented in the following paragraphs are obtained with the following implementation. The model comparison overhead routine is written in Python 3.6. The compared discrete kernels are implemented within the framework of a GPflow [83], a Python based toolbox for GP-based modeling relying on the Tensorflow machine learning platform [1] (version 1.13). The GP training is performed with the help of a Bounded Limited memory Broyden - Fletcher - Goldfarb – Shanno (L-BFGS-B) algorithm [24].

3.7.2 Branin function

The first analytical benchmark function to be considered is a modified version of the Branin function [42] characterized by two continuous variables and two discrete variables, each one with 2 levels, thus resulting in overall 4 discrete categories. The analytical definition of this mixed-variable Branin function is the following:

$$f(x_1, x_2, z_1, z_2) = \begin{cases} h(x_1, x_2) & \text{if } z_1 = 0 \text{ and } z_2 = 0 \\ 0.4h(x_1, x_2) + 1.1 & \text{if } z_1 = 0 \text{ and } z_2 = 1 \\ -0.75h(x_1, x_2) + 5.2 & \text{if } z_1 = 1 \text{ and } z_2 = 0 \\ -0.5h(x_1, x_2) - 2.1 & \text{if } z_1 = 1 \text{ and } z_2 = 1 \end{cases} \tag{3.74}$$

where:

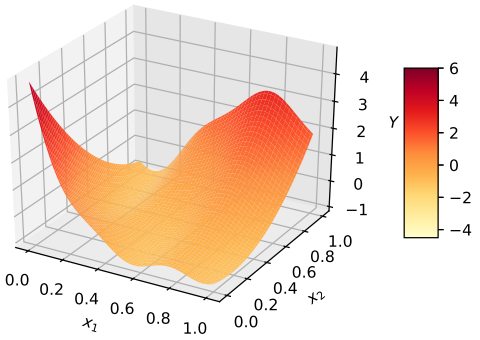
$$\begin{aligned} h(x_1, x_2) =& (((15x_2 - \frac{5}{4\pi^2}(15x_1 - 5)^2 + \frac{5}{\pi}(15x_1 - 5) - 6)^2 + \\ & 10\left(1 - \frac{1}{8\pi}\right)\cos(15x_1 - 5) + 10) - 54.8104)/51.9496 \end{aligned} \tag{3.75}$$

with

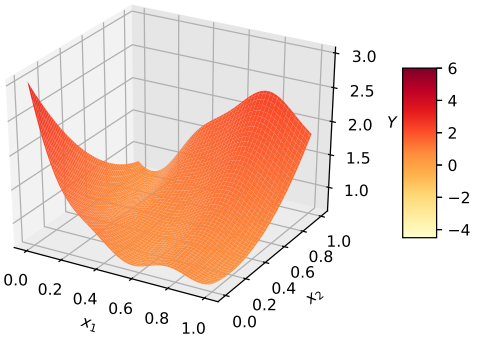
$$x_1 \in [0,1], \quad x_2 \in [0,1], \quad z_1 \in \{0,1\}, \quad z_2 \in \{0,1\}$$

For illustrative purposes, the responses associated to each category of the mixed-variable Branin function are presented in Figure 3.9. By analyzing the function definition, two particular characteristics which may influence the modeling performance of the various kernels can be highlighted. The first noticeable characteristic is the presence of an negative correlation between the responses characterized by the 2 levels of the variable z_1: $z_1 = 0$ and $z_1 = 1$ (*i.e.,* the global trends present opposite variations with respect to the continuous variables). Secondly, although the function depends on 2 independent discrete variables, it can be noticed that its categories are defined as a function of the discrete level combinations, rather than as a function of the discrete levels independently.

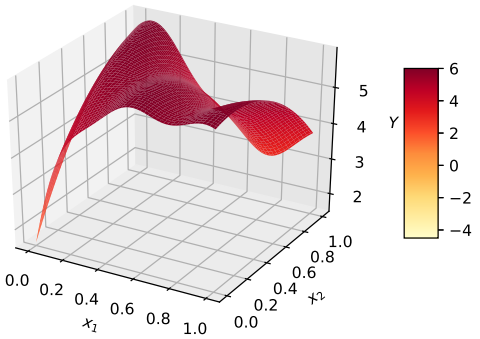
The modeling performance benchmark obtained for the mixed-variable Branin function are provided in Figure 3.10. These results are obtained over 20 different training data sets of 20, 40 and 80 samples (*i.e.*, 5, 10 and 20 samples per discrete category). Overall, it can be seen that for this test-case most of the considered mixed-variable surrogate modeling methods provide a better modeling accuracy than the independent CW GP. The only exceptions being the homoscedastic and heteroscedastic category-wise CS kernels when insufficient data is provided (*i.e.,* the 20



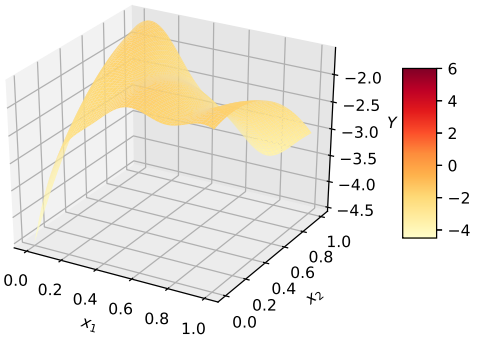
Category 1: $z_1 = 0, z_2 = 0$



Category 2: $z_1 = 0, z_2 = 1$



Category 3: $z_1 = 1, z_2 = 0$



Category 4: $z_1 = 1, z_2 = 1$

FIGURE 3.9: The 4 discrete categories of the Branin function.

training data samples case). It can also be noticed that for all 3 training data set sizes, both variants of the level-wise CS and LV kernels provide nearly identical results. This is due to the fact that in case the modeled variable is only characterized by 2 levels, both kernel parameterizations rely on a single hyperparameter to compute the covariance between the levels and therefore yield very similar results. The slight difference between the results obtained with the 2 methods is due to the different hyperparameter initialization in the GP model training. The same similarity is not found for CS and LV category-wise variants, as in this case the discrete kernels model 4 levels instead of 2. A further noticeable trend in the presented results is that both level-wise and category-wise CS and LV kernels tend to provide worse results when compared to the rest of the considered methods. This can be explained by the fact that these kernels can not return negative covariance values, which makes it so that they cannot properly model the negative correlation trends characterizing the levels of z_1. Furthermore, the results show that overall the heteroscedastic variants of the considered kernels tend to produce better results with respect to their homoscedastic counterparts. This is closely related to the heteroscedastic nature of the mixed-variable Branin function which is easily noticeable from its analytical definition. Finally, the results show that for large enough training data sets, the best performing kernels in terms of modeling accuracy are the HS_C, HS_C_He and the CN_C. Additionally to the previous considerations, these results can be explained by the fact that when sufficient data is provided, the category-wise modeling of the discrete variables allows to better capture the fact that each

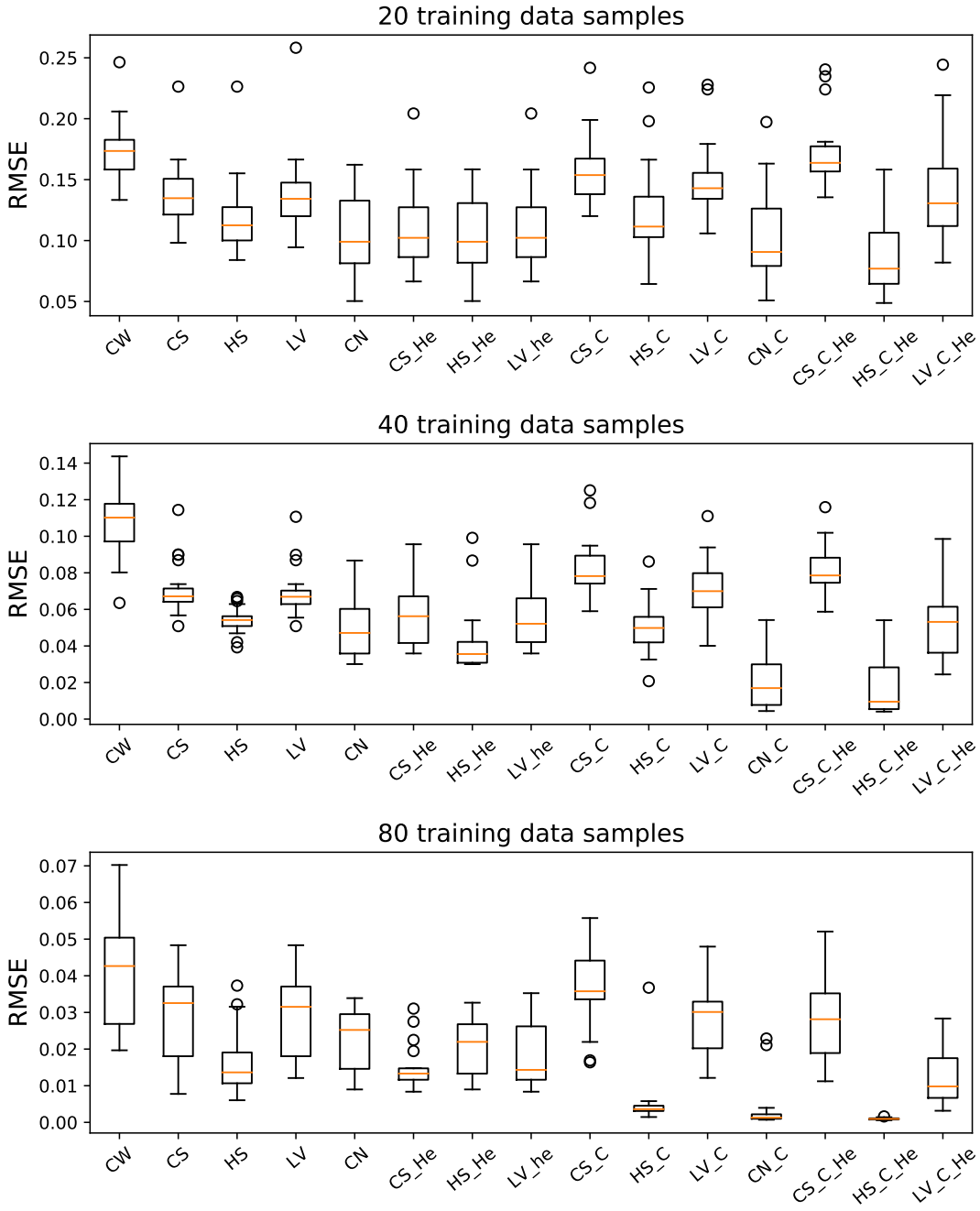


FIGURE 3.10: Comparison of various discrete kernels modeling performance on the mixed-variable Branin function for various training data set sizes over 20 repetitions.

category of the considered mixed-variable Branin is defined as a function of the discrete level combinations, rather than as a function of the discrete levels independently. This difference is less noticeable for smaller data sets, in which case the information provided to the GP is not sufficient to properly model each function category.

3.7.3 Augmented Branin function

The second analytical benchmark function to be considered is an augmented (*i.e.,* larger dimension) version of the previously described Branin function, characterized by 10 continuous variables and 2 discrete variables, each one with 2 levels, thus resulting in 4 discrete categories. The main purpose of this test-case is to assess the dependency of the various discrete kernel parameterizations performance with respect to the size of the continuous design space of the considered problem. In practice, the augmented Branin function is defined as follows:

$$f(x_1,...,x_{10},z_1,z_2) = \begin{cases} \tilde{h}(x_1,...,x_{10}) & \text{if } z_1 = 0 \text{ and } z_2 = 0 \\ 0.4\tilde{h}(x_1,...,x_{10}) + 1.1 & \text{if } z_1 = 0 \text{ and } z_2 = 1 \\ -0.75\tilde{h}(x_1,...,x_{10}) + 5.2 & \text{if } z_1 = 1 \text{ and } z_2 = 0 \\ -0.5\tilde{h}(x_1,...,x_{10}) - 2.1 & \text{if } z_1 = 1 \text{ and } z_2 = 1 \end{cases} \tag{3.76}$$

where:

$$\tilde{h}(x_1,...,x_{10}) = \frac{h(x_1,x_2) + h(x_3,x_4) + h(x_5,x_6) + h(x_7,x_8) + h(x_9,x_{10})}{5} \tag{3.77}$$

with:

$$\begin{aligned} h(x_i,x_j) =& (((15x_j - \frac{5}{4\pi^2}(15x_i - 5)^2 + \frac{5}{\pi}(15x_i - 5) - 6)^2 + \\ & 10\left(1 - \frac{1}{8\pi}\right)\cos(15x_i - 5) + 10) - 54.8104)/51.9496 \end{aligned} \tag{3.78}$$

and

$$x_1,...,x_{10} \in [0,1], \quad z_1 \in \{0,1\}, \quad z_2 \in \{0,1\}$$

For this benchmark function, the testing is performed with training data sets of 80, 160 and 320 samples (*i.e.*, 20, 40 and 80 samples per discrete category). The results are provided in Figure 3.11. An analysis of these results shows that for small training data sets (*i.e.,* 80 samples), no real difference between the various considered kernels can be highlighted due to the insufficient information to train the GP model. For larger training data sets, instead, it is shown that mixed-variable GP yield overall better results when compared to the independent CW GP, with the exception of the category-wise modeling with a CS kernel. Furthermore, similarly to what is shown for the previously discussed mixed-variable Branin function, heteroscedastic approaches tend to provide a better modeling accuracy, due to the ability to better capture the heteroscedastic trends of the considered function. Finally, it can be seen that if sufficient data is provided, a category-wise approach of the mixed-variable surrogate modeling yields the best results, if combined with heteroscedastic kernels (*i.e.,* CN_C and HS_C_He). This is related to the fashion in which the categories of this particular benchmark function are constructed, as is discussed in the previous paragraphs. In general, it is shown that the relative performance of the considered kernel parameterizations does not significantly vary when the size of the continuous search space is increased. The main noticeable difference is that larger training data sets are required in order to obtain the same range of modeling accuracy, which can be explained by the larger number of hyperparameters required to characterize the continuous kernels.

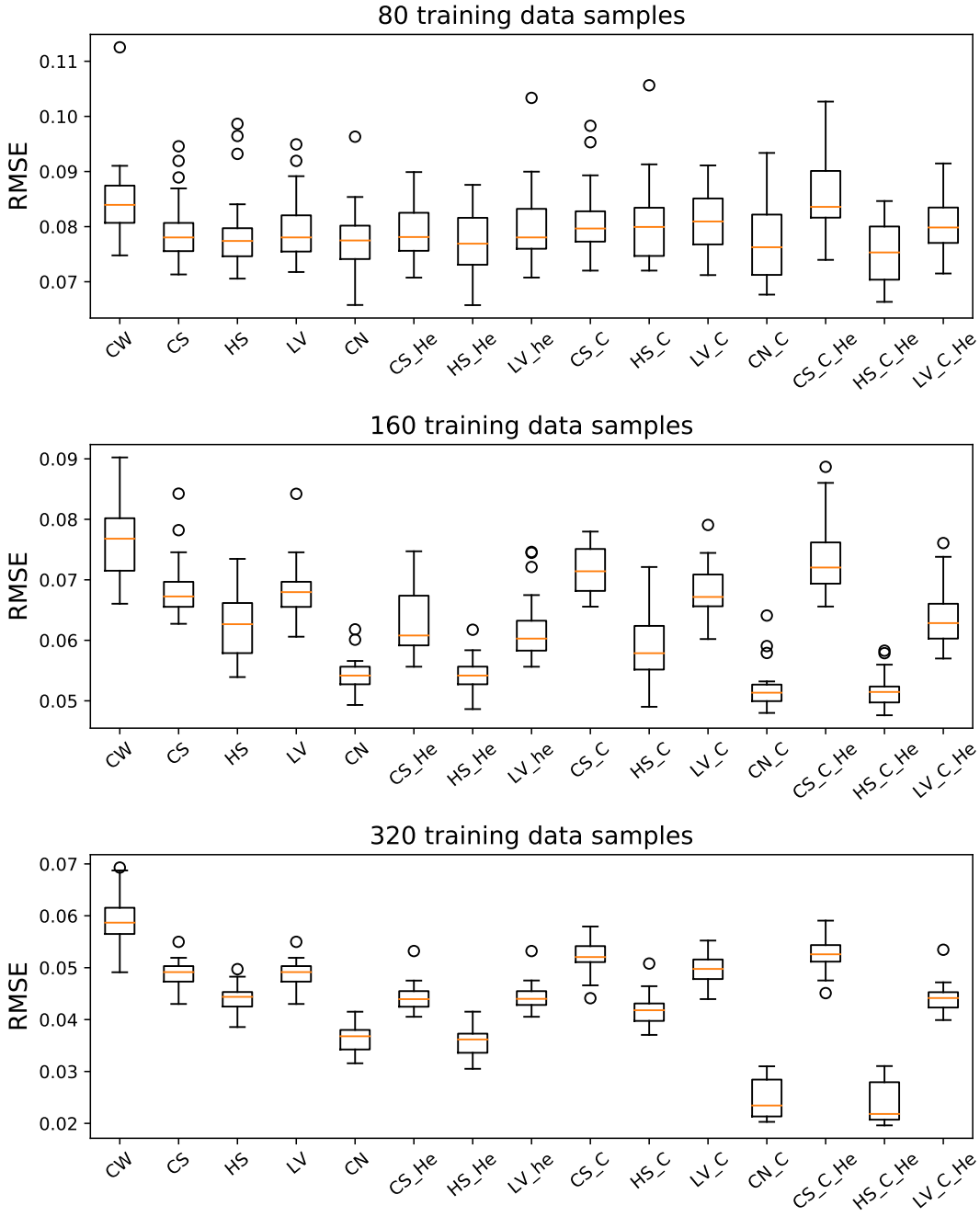


FIGURE 3.11: Comparison of various discrete kernels modeling performance on the augmented mixed-variable Branin function for various training data set sizes over 20 repetitions.

3.7.4 Goldstein function

The third analytical benchmark function to be considered in this work is a modified mixed-variable variant of the Goldstein [100] function characterized by 2 continuous variables and 2 discrete variables, each one with 3 levels, thus resulting in 9 discrete categories. This mixed-variable Goldstein function is defined as follows:

$$f(x_1, x_2, z_1, z_2) = h(x_1, x_2, x_3, x_4) \tag{3.79}$$

with

$$x_1 \in [0, 100], \quad x_2 \in [0, 100], \quad z_1 \in \{0, 1, 2\}, \quad z_2 \in \{0, 1, 2\}$$

The values of x_3 and x_4 are determined as a function of z_1 and z_2 according to the relations defined in Table 3.2.

	$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$z_2 = 0$	$x_3 = 20, x_4 = 20$	$x_3 = 50, x_4 = 20$	$x_3 = 80, x_4 = 20$
$z_2 = 1$	$x_3 = 20, x_4 = 50$	$x_3 = 50, x_4 = 50$	$x_3 = 80, x_4 = 50$
$z_2 = 2$	$x_3 = 20, x_4 = 80$	$x_3 = 50, x_4 = 80$	$x_3 = 80, x_4 = 80$

TABLE 3.2: Characterization of the Goldstein function discrete categories

$$\begin{aligned} h(x_1, x_2, x_3, x_4) = \; & 53.3108 + 0.184901 x_1 - 5.02914 x_1^3 10^{-6} + 7.72522 x_1^4 10^{-8} - \\ & 0.0870775 x_2 - 0.106959 x_3 + 7.98772 x_3^3 10^{-6} + \\ & 0.00242482 x_4 + 1.32851 x_4^3 10^{-6} - 0.00146393 x_1 x_2 - \\ & 0.00301588 x_1 x_3 - 0.00272291 x_1 x_4 + 0.0017004 x_2 x_3 + \\ & 0.0038428 x_2 x_4 - 0.000198969 x_3 x_4 + 1.86025 x_1 x_2 x_3 10^{-5} - \\ & 1.88719 x_1 x_2 x_4 10^{-6} + 2.50923 x_1 x_3 x_4 10^{-5} - \\ & 5.62199 x_2 x_3 x_4 10^{-5} \end{aligned} \tag{3.80}$$

This benchmark function presents similar continuous and discrete design space dimensions to the previously discussed mixed-variable Branin function, however, a few key characteristics differ. First of all, although both functions depend on the same number of discrete variables, the Goldstein functions presents a larger number of levels per discrete variable, thus resulting in more than twice as many categories. On the other hand, the Goldstein function can be expected to be less challenging to model due to the absence of negative correlations between discrete levels as well as the presence of a more homoscedastic global trend when compared to the Branin function. Finally, it can also be noticed that each discrete variable has an independent influence on the definition of each problem category, which is not the case for the two Branin function variants previously considered. For this benchmark function, the testing is repeated with data sets of 27, 72 and 135 samples (*i.e.*, 3, 8 and 15 samples per discrete category). The results obtained for the modeling of the Goldstein function with the considered discrete kernels are provided in Figure 3.12. As for the previous test-cases, the independent CW GP approach tends to provide considerably worse modeling performance, most notably for larger data sets, due to lower amount of exploited information. A second noticeable trend which can be identified in the results is that for small sized training data sets (*i.e.,* 27 data samples), the category-wise kernels tend to provide a worse modeling accuracy when compared to the ones based on a level-wise approach. This can be explained by the relatively large number of categories with respect to the available data, which can be challenging to model independently. Additionally, it can also be seen that the heteroscedastic

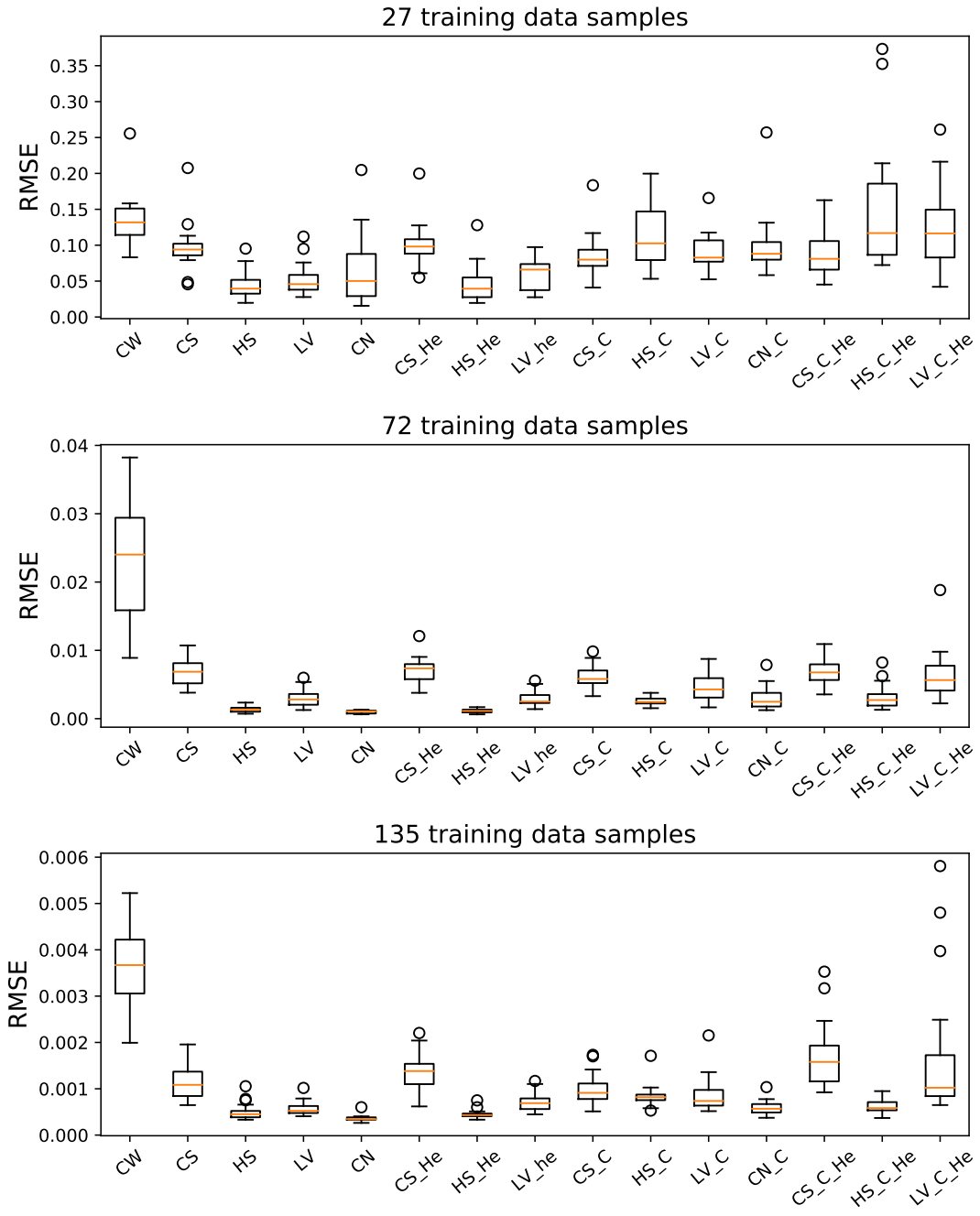


FIGURE 3.12: Comparison of various discrete kernels modeling performance on the mixed-variable Goldstein function for various training data set sizes over 20 repetitions.

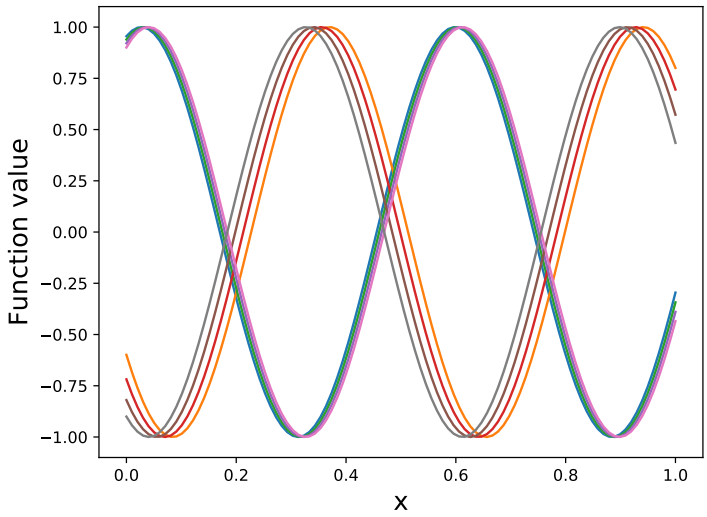


FIGURE 3.13: Categories of the 4th modeling analytical benchmark.

variants of the considered kernels do not seem to provide a sizable advantage in terms of modeling performance with respect to their homoscedastic counterpart. As previously mentioned, this is due to the fact that the Goldstein function is not properly heteroscedastic, and by consequence, considering a heteroscedastic model results in a number of unnecessary hyperparameters to be tuned. Finally, for large enough training data sets the LV, HS and CN kernel parameterizations tend to provide the most promising modeling results, with a slightly better performance for the latter two.

3.7.5 Analytical benchmark N.4

The fourth analytical benchmark function which is considered in this chapter is an adaptation of a function proposed in [108], characterized by a single continuous variable and 2 discrete variables, presenting respectively 4 and 2 levels, for a total of 8 categories. The considered function is defined as follows:

$$f(x, z_1, z_2) = \begin{cases} \cos(7\pi\frac{x}{2} - \frac{z_1}{20}) & \text{if } z_2 = 0 \\ \cos(7\pi\frac{x}{2} + (0.4 + \frac{z_1}{15})\pi - \frac{z_1}{20}) & \text{if } z_2 = 1 \end{cases} \tag{3.81}$$

with $x \in [0,1]$, $z_1 \in \{6,7,8,9\}$ and $z_2 \in \{0,1\}$. For illustrative purposes, the 8 categories of the function defined above are plotted in Figure 3.13. This benchmark is expected to be characterized by homoscedastic trends combined with a negative covariance between the levels of the discrete variable z_2. For this function, the testing is repeated with data sets of 40, 80 and 120 samples (*i.e.*, 5, 10 and 15 samples per discrete category). The obtained modeling results are provided in Figure 3.14. Overall, it can be seen that most of the considered discrete kernels provide a considerably better modeling accuracy when compared to the independent category-wise GP, the only exception being the homoscedastic and heteroscedastic category-wise CS and LV kernels, which tend to poorly model the negative correlation trends. The results also show that due to the relatively homoscedastic nature of this particular test-case, the heteroscedastic variants of the considered discrete kernels do not yield sizable advantages in terms of modeling accuracy.

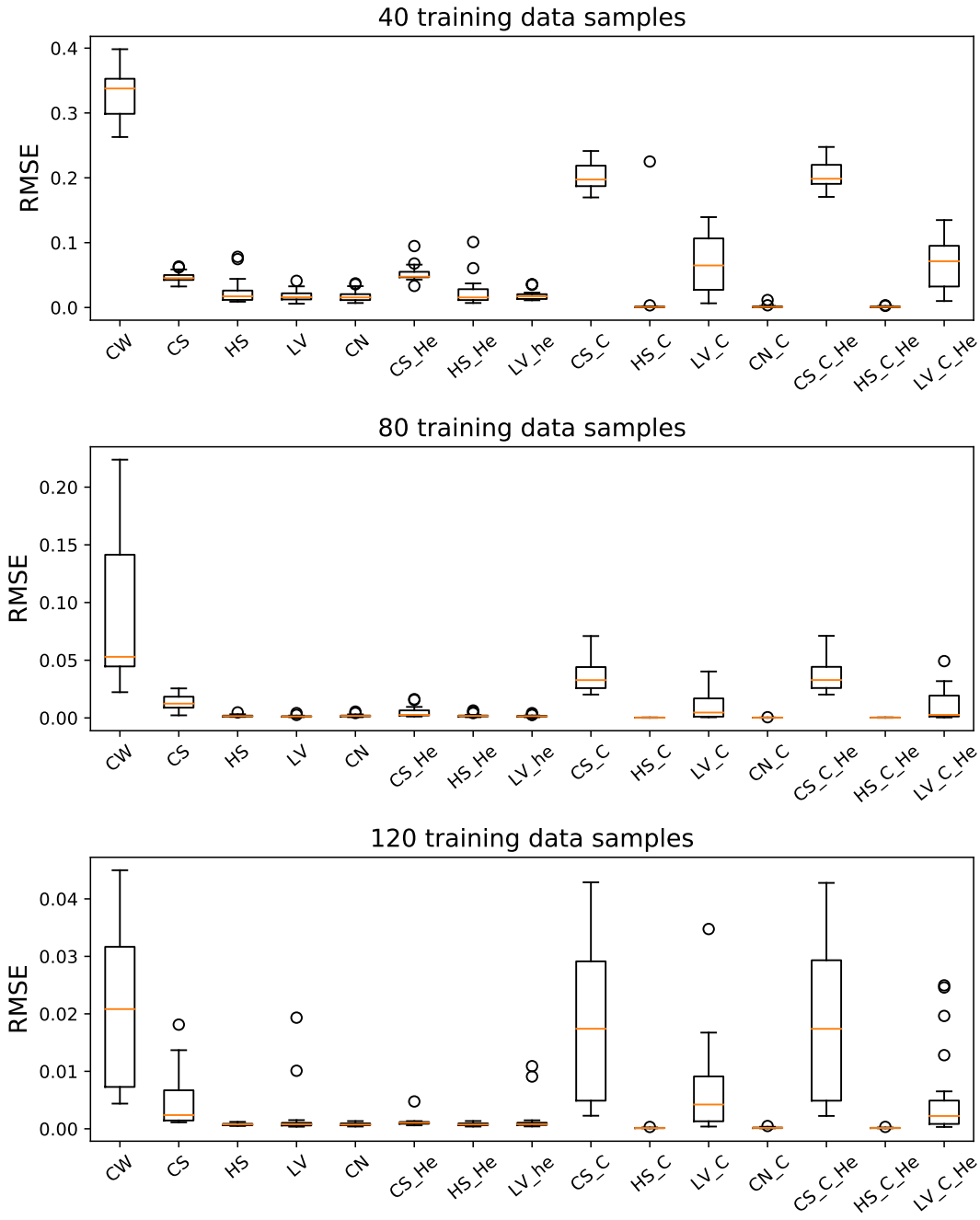


FIGURE 3.14: Comparison of various discrete kernels modeling performance on the fourth analytical benchmark function for various training data set sizes over 20 repetitions.

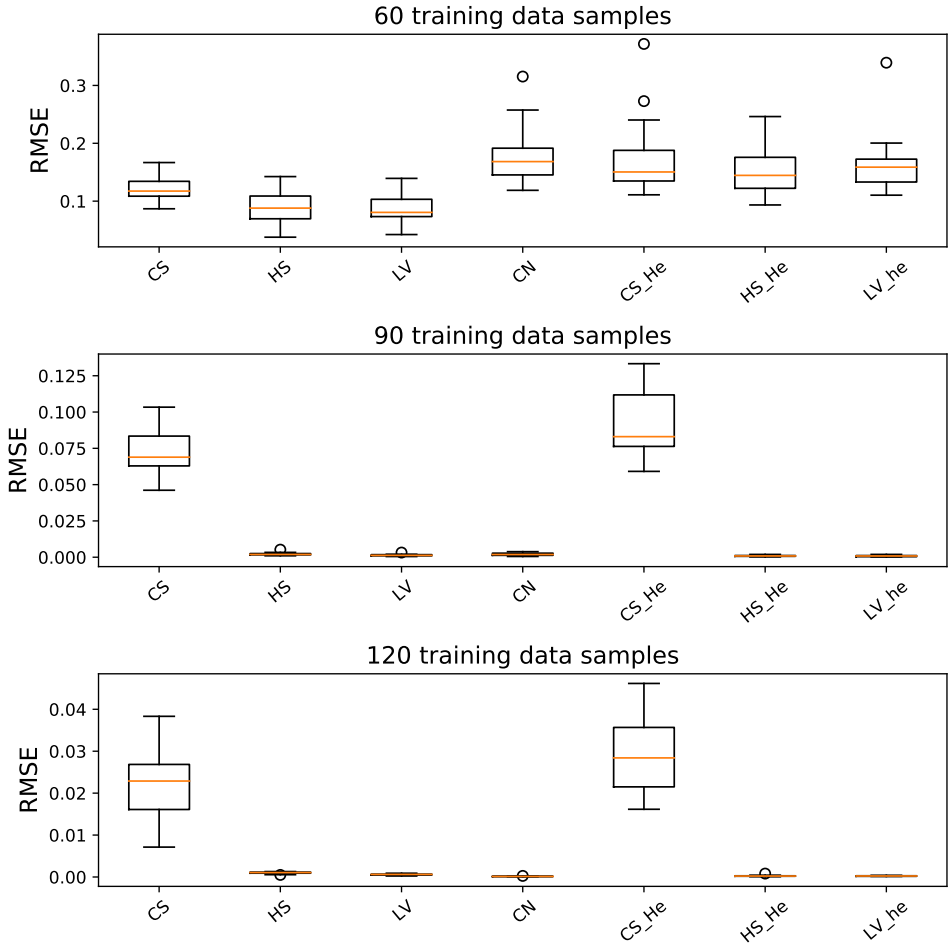


FIGURE 3.15: Comparison of various discrete kernels modeling performance on the fifth analytical benchmark function for various training data set sizes over 20 repetitions.

3.7.6 Analytical benchmark N.5

The fifth and final analytical benchmark function which is considered in this chapter is an adaptation of a function proposed in [29], characterized by 5 continuous variables and 5 discrete variables. Each one of the discrete variables is associated to 3 levels, which results in a total of 243 categories. This test-function is defined as follows:

$$f(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{5} \frac{x_i(z_{6-i}-2)}{80} \prod_{i=1}^{5} \cos\left(\frac{x_i}{\sqrt{i}}\right) \sin\left(\frac{50(z_{6-i}-2)}{\sqrt{i}}\right) \tag{3.82}$$

with $\mathbf{x} = \{x_1,\dots,x_5\} \in [0,1]^5$, and $z_i \in \{1,2,3\}$ for $i = \{1,...,5\}$. The mixed-variable function defined above is characterized by a large number of categories relatively to the total dimension of its design space. As a consequence, both the independent category-wise GP and category-wise discrete kernels cannot reasonably be considered for this benchmark, as the number of training data samples would be smaller than the total number of categories, and by extension also lower than the number of hyperparameters to be tuned. For this benchmark the testing is repeated with data sets of 60, 90 and 120 samples. The results obtained with the considered level-wise discrete kernels are provided in Figure 3.15. The results show that for small training data sets, not enough information is provided in order to properly train heteroscedastic kernels, and by consequence the homoscedastic ones yield slightly better results. For larger training data sets,

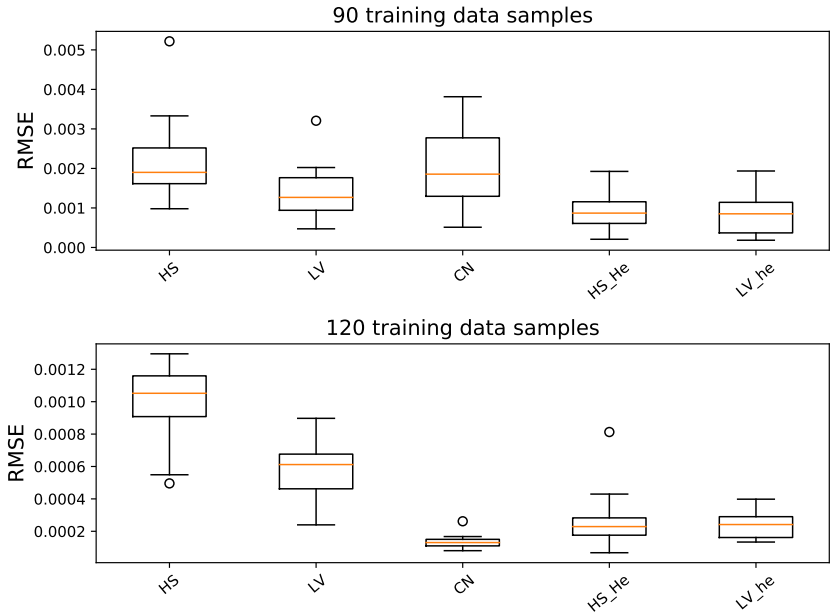


FIGURE 3.16: Comparison of various discrete kernels modeling performance on the fifth analytical benchmark function for various training data set sizes over 20 repetitions.

instead, both variants of the CS kernel are too simplistic and thus unable to properly capture the trends of the considered function. In order to be able to distinguish the differences in performance between the remaining kernel parameterizations, the same results of Figure 3.15 are presented in Figure 3.16 without the 2 CS kernel variants, It can then be seen that when sufficient information is provided to the model, the heteroscedastic kernels provide the most accurate modeling of the considered function, with the best results being associated to the CN kernel.

3.7.7 Propulsion performance simulation

In order to better assess the surrogate modeling performance of the discrete kernels discussed in this chapter, 2 aerospace design related test-cases are considered as benchmarks. The first representative test-case is a simulation of the combustion performance of a launcher engine. More specifically, the modeled variable is the specific impulse I_{SP} provided by the engine. The I_{SP} is modeled as a function of the reductant to oxidant ratio O_F, the combustion chamber pressure P_c, the nozzle area ratio ϵ (defined as the ratio between the nozzle throat diameter and the nozzle exit diameter) and the type of reductant and oxidant. The first three variables characterizing the I_{SP} are continuous, while the type of reductant and oxidant are discrete choices which can be represented with the use of discrete variables. A summary of the variables characterizing the problem is provided in Table 3.3. As an illustrative example, a visual representation of a liquid propulsion engine, namely the Prometheus engine, is provided in Figure 3.17.

Variable	Nature	Min	Max	Levels
O_F	continuous	2.5	5.5	[-]
P_c [bar]	continuous	20	80	[-]
ϵ	continuous	10	60	[-]
Oxidant	discrete	[-]	[-]	O2, N204, F2, H2O2
Reductant	discrete	[-]	[-]	CH4, N2H4, JP-4, H2

TABLE 3.3: Variables characterizing the combustion performance simulation test-case

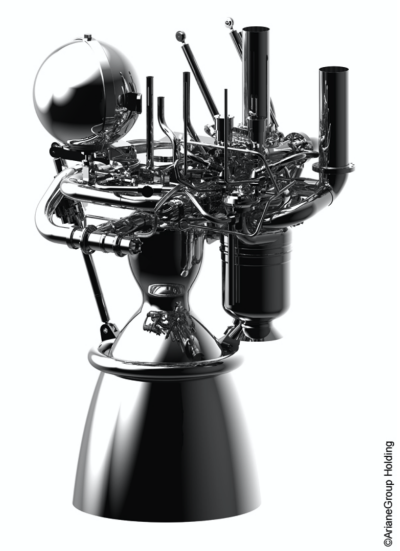


FIGURE 3.17: Prometheus engine, courtesy of ArianeGroup.

Although this problem is theoretically characterized by a total of 16 categories, it is important to note that not all the combinations of reductant and oxidant can realistically be simulated and therefore only 7 categories are modeled. The combustion simulations necessary to create the training data sets are performed by using the thermo-chemical simulation software 'Chemical Equilibrium with Applications' (CEA) created by NASA [84]. This software simulates the combustion of gases in a combustion chamber and their expansion in an engine nozzle. The conditions for chemical motion equilibrium are stated in terms of Gibbs and Helmholtz energies [71] or the maximization of the entropy. The system of equations which characterizes the equilibrium and describes its composition is non-linear and therefore iterative methods (such as the Newton-Rhapson method [20]) are used to solve it. In Figure 3.18, the I_{SP} profiles for the 7 considered combinations of reductant and oxidant are shown as a function of the nozzle ratio ε and the reductant to oxidant ratio O_F.

The results obtained when modeling the launcher engine specific impulse are provided in Figure 3.19. These results are obtained over 20 different training data sets of 21, 56 and 105 samples (*i.e.*, 4, 8 and 15 samples per discrete category). As for the previous test-cases, it is shown that mixed-variable modeling provides a more accurate prediction of the modeled function values with respect to independent continuous CW GP. Furthermore, it can be seen that for small training data sets (*i.e.,* 21 samples), heteroscedastic kernels tend to yield worse results when compared to the homoscedastic ones due to the larger number of hyperparameters to train with insufficient data. However, it can be noticed that when sufficient data is provided (*i.e.,* 105 data samples), the relative difference in performance between the compared surrogate models diminishes. This can be explained by the fairly smooth and linear trends of the modeled function with respect to the design variables (as is shown in Figure 3.18) that often characterizes physical phenomena, which result in an easier modeling process.

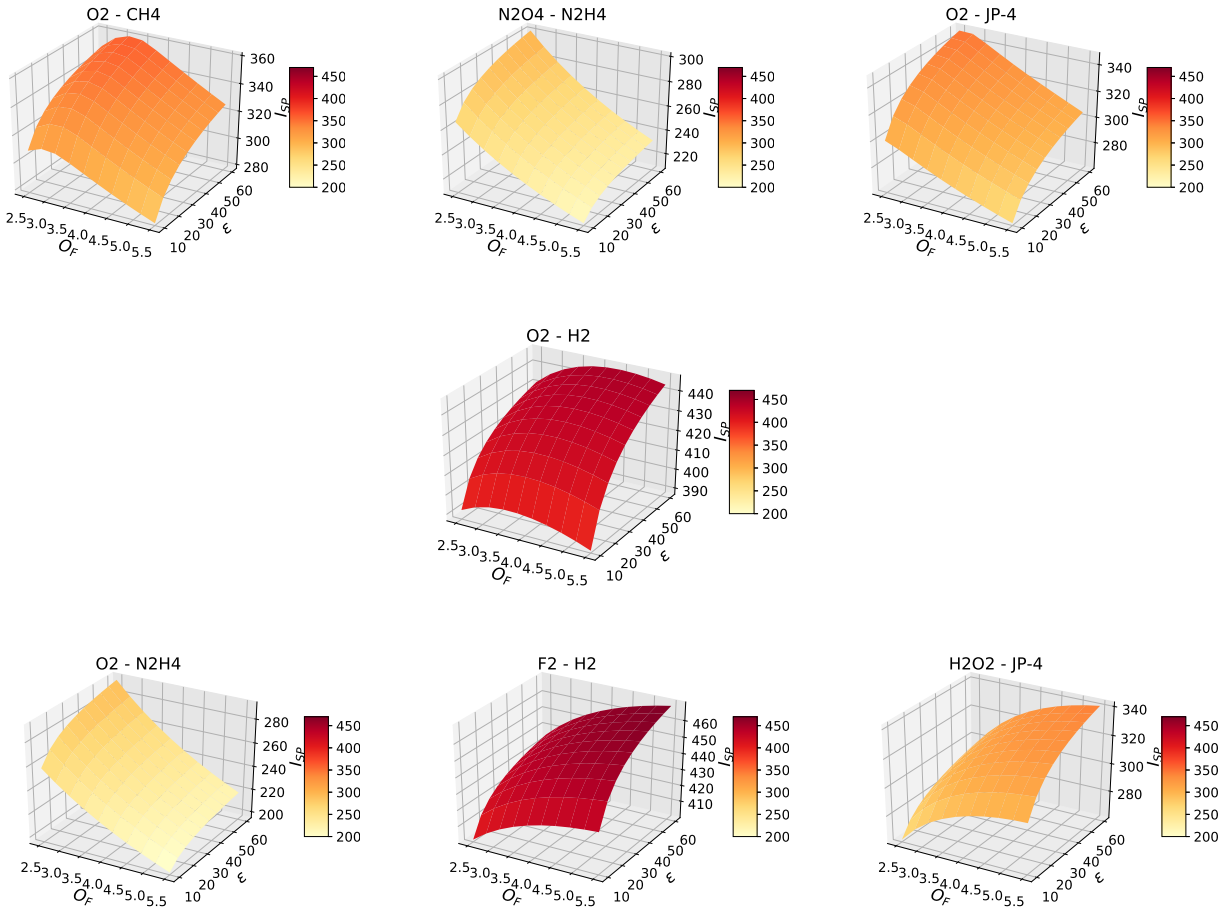


FIGURE 3.18: Specific impulse of a launcher engine as a function of the nozzle ratio ϵ and the reductant to oxidant ratio O_F for various combinations of oxidant and reductant.

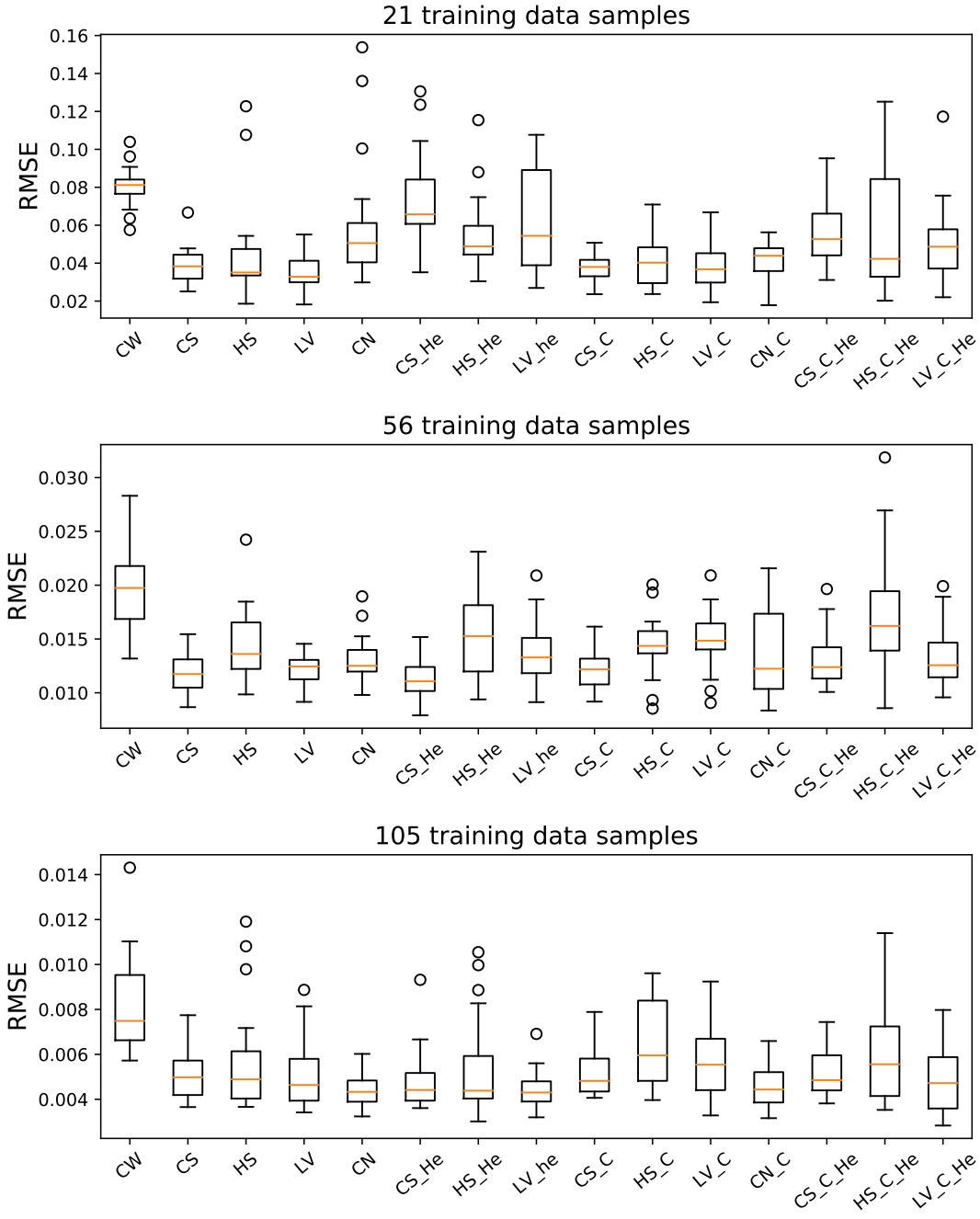


FIGURE 3.19: Comparison of various discrete kernels modeling performance on the launcher engine specific impulse test-case for various training data set sizes over 20 repetitions.

3.7.8 Thrust frame structural analysis

The second aerospace design test-case that is considered is the modeling of a launcher thrust frame stiffening, commonly performed for preliminary sizing purposes. More specifically, the Ariane 6 aft bay structural characteristics are analyzed. The thrust frame is located at the bottom of the launcher first stage and has the purpose of withstanding the weight of the launcher as well as the thrust of the two solid rocket boosters during the lift-off phase. A schematic representation of the Ariane 6 aft bay and its location within the launch vehicle system is provided in Figure 3.20. This structure is composed of a cylindrical outer skin, stiffened by frames and stringers, and an inner body comprising three major frames and an inner skin.

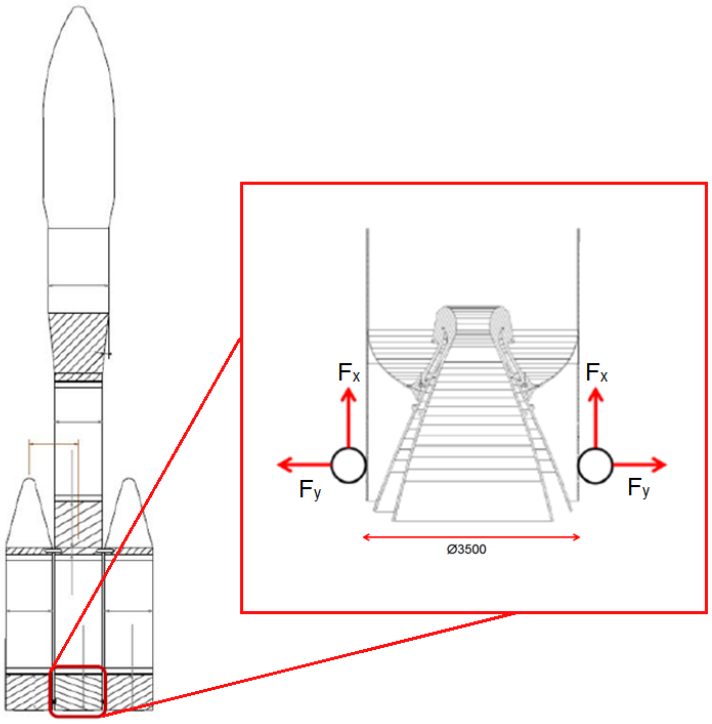


FIGURE 3.20: Ariane 6 PPH launcher aft bay.

The static loads that are considered for this simulation are the longitudinal and lateral thrust of the two solid rocket boosters (see Figure 3.20). The boundary conditions are defined by modeling the first stage composite bottom skirt clamped to the upper interface of the thrust frame. In this test-case, two different parameters of the considered system are modeled: the maximum Von Mises stress on the inner skin of the structure [40] and the upper interface longitudinal over-flux. These are modeled as a function of the inner and outer skins thicknesses t_i, t_o of the 6 regions in which the thrust frame is divided as well as of the number of stringers N_s and the number of frames N_f. The 12 thicknesses are continuous variables while the number of stringers and frames are discrete variables, each one characterized by 3 levels, thus resulting in a total of 9 categories. For illustrative purposes, structural responses on the entire thrust frame structure considering the maximum Von Mises stress and the over-flux are provided in Figure 3.21. For the sake of simplicity, the same range is considered for all the thicknesses variables characterizing the problem. More specifically, the minimum and maximum bounds are 1 and 30 mm, respectively. A summary of the variables characterizing the studied problem is provided in Table 3.4.

In order to generate the training and testing data sets for this analysis, the MSC Nastran Finite Element Method (FEM) software [82] is used. In practice, a separate finite element model is created for every considered category of the problem (*i.e.*, for every combination of the number of stringers and frames) due to the need of a distinct meshing for every configuration. A number of

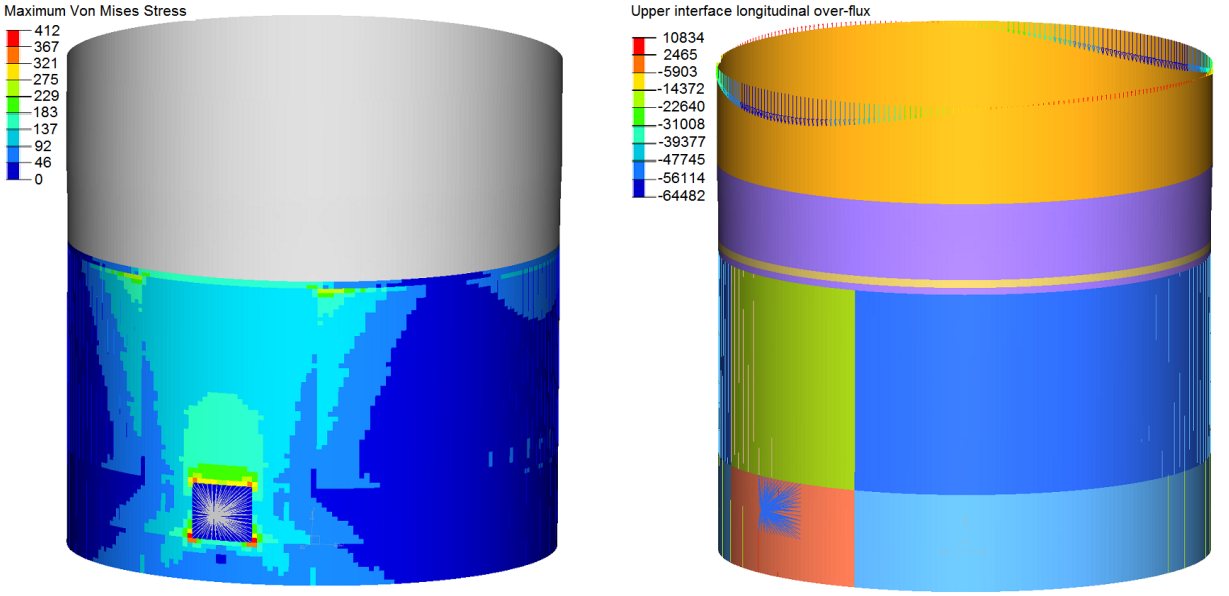


FIGURE 3.21: Examples of structural responses on the entire thrust frame structure. On the left figure the maximum Von Mises stress is illustrated, while on the right figure the upper interface longitudinal over-flux is shown.

Variable	Nature	Min	Max	Levels
$t_{i_{1,6}}$ [mm]	continuous	1	30	[-]
$t_{o_{1,6}}$ [mm]	continuous	1	30	[-]
N_s	discrete	[-]	[-]	36, 72, 144
N_f	discrete	[-]	[-]	2, 4, 8

TABLE 3.4: Variables characterizing the thrust frame structural analysis test-case

static FEM analyses is then performed with varying inner and outer skins thicknesses, according to the values present in the data sets, and using the finite element model corresponding to the category the considered sample belongs to. The lift-off conditions are simulated by considering two aligned vertical loads F_x and two opposing horizontal loads F_y, applied on the side of the thrust frame, as illustrated in Figure 3.20.

Due to the larger computation cost of the static load simulation when compared to the analytical test-cases, the modeling performance benchmark is only performed over 10 repetitions, with each training data set containing 135 samples. The results obtained when performing the surrogate modeling of the maximum inner skin Von Mises stress and the upper interface longitudinal over-flux on the thrust frame are shown in the upper and lower plots of Figure 3.22, respectively. The results show that overall, relying on mixed-variable kernels allows to considerably reduce the modeling error if compared to independent CW GP. It can also be noticed that category-wise mixed-variable kernels yield worse results with respect to a level-wise modeling, due to the nature of the modeled functions as well as the relatively small data sets size. No significant difference in performance between CS/LV kernels and HS/CN kernels can be noticed, which can be explain by a lack of negative correlation trends between the discrete levels and/or categories of the considered problem. Finally, heteroscedastic and homoscedastic kernels yield similar results, which again suggests the absence of considerably heteroscedastic trends in the modeled function.

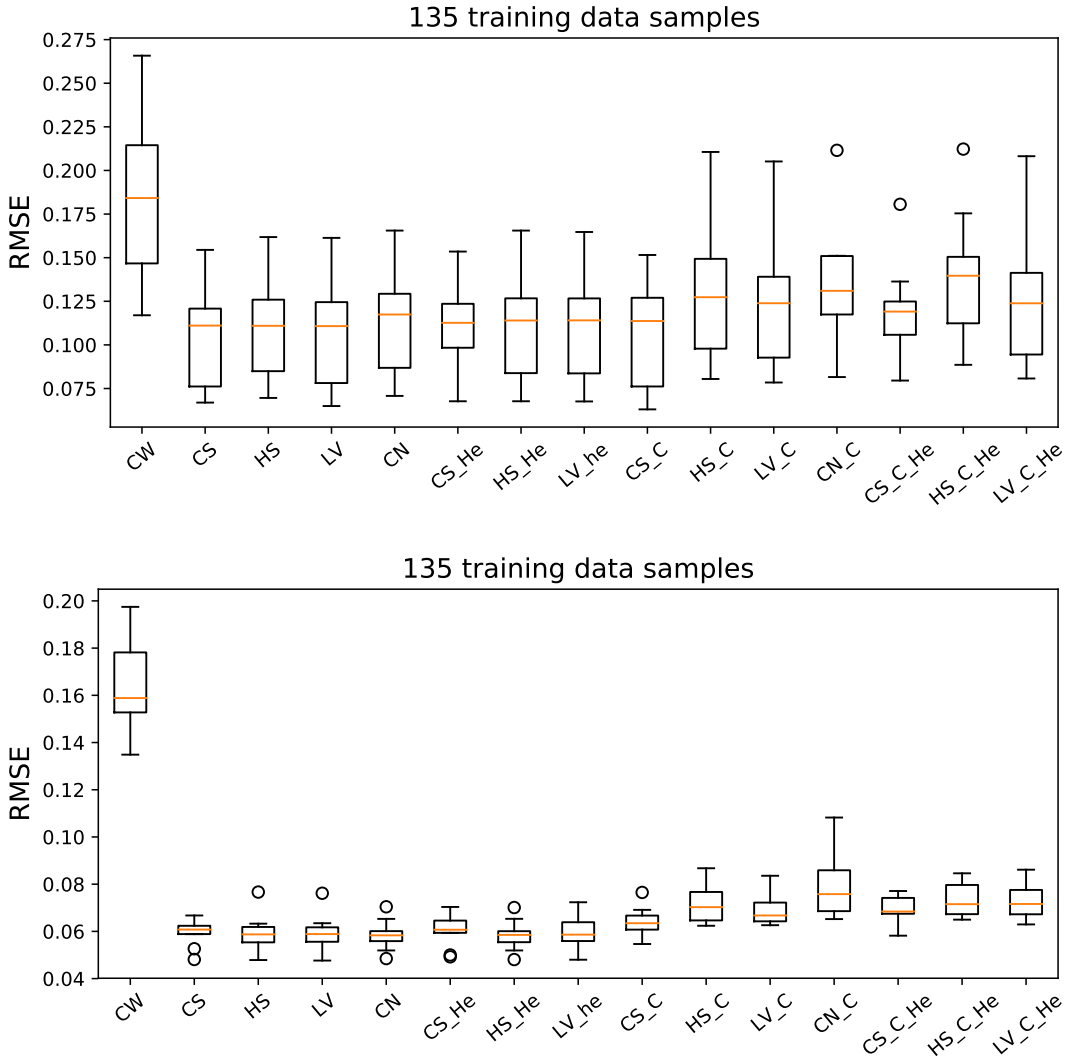


FIGURE 3.22: Comparison of various discrete kernels modeling performance on the thrust frame structural analysis test-case. The modeled values are the maximum inner skin Von Mises stress (top) and the upper interface longitudinal over-flux (bottom) over 10 repetitions.

3.8 Error model

Within the context of function modeling, the main property which is usually considered is the modeling accuracy, *i.e.,* the difference between the actual and the predicted function value, which can be estimated through criteria such as the RMSE. However, within the context of GP based surrogate-model based design optimization, the validity of the error model (*i.e.,* variance prediction $\hat{s}^2(\cdot)$) is also relevant, as it drives the exploration aspect of the optimization process. In other words, for a GP based BO to be efficient, it is not sufficient for the prediction to be accurate, the error model must also be coherent. As a measure of the error model coherence, the Mean Negative test Log-Likelihood (MNLL) is considered. Similarly to the RMSE, this measure is computed on a test data set of N samples as:

$$\text{MNLL} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{1}{\sqrt{2\pi\hat{s}^2(\mathbf{x}^i,\mathbf{z}^i)}}\exp\left(-\frac{\left(y(\mathbf{x}^i,\mathbf{z}^i)-\hat{y}(\mathbf{x}^i,\mathbf{z}^i)\right)^2}{2\hat{s}^2(\mathbf{x}^i,\mathbf{z}^i)}\right)\right) \tag{3.83}$$

and it represents the (negative) likelihood of predicting the exact value of the data set samples with the considered GP model prediction (in terms of both mean prediction and associated variance). As for the RMSE, lower values of the MNLL tend to characterize a better performing surrogate model uncertainty estimation. For illustrative purposes, the error model produced by the various discrete kernels considered in this chapter is compared on the most representative analytical and design related test-cases.

First the mixed-variable Branin function is considered. As for the modeling accuracy benchmark, the test is repeated 20 times over data sets of 20, 40 and 80 samples and validated on a data set of a 1000 samples. The obtained results are provided in Figure 3.23.

Overall, the results show a similar relative performance between the various kernels as for the prediction benchmark. The main difference is represented by the independent CW approach which tends to provide an error model accuracy comparable to most of the considered kernels for small sized data sets (*i.e.,* 20 and 40 data samples). A Slightly worse relative performance of the heteroscedastic hypersphere decomposition kernel can also be noticed.

The second considered benchmark is the thrust frame structural analysis. In this case, the test is repeated 10 times over data sets of 135 data samples. The obtained results for the modeling of the maximum Von Mises stress on the inner skin of the structure and the upper interface longitudinal over-flux are provided in the top and bottom parts of Figure 3.24, respectively.

Also for this benchmark, the results show that the MNLL provided by the independent CW approach, relatively to the rest of the discrete kernels, is considerably better than its prediction RMSE. Furthermore, it can also be noticed that differently from the modeling benchmark, most of the category-wise approaches overall tend to provide a more accurate error model when compared to the level-wise approaches. This might be explained by the fact that sufficient data is provided in order to properly optimize the variance associated to each category of the problem.

3.9 Result synthesis

In the previous section, the different discrete kernel parameterizations described in Section 3.5 are tested on several analytical and engineering related benchmark functions. Overall, it is shown that mixed-variable GP tend to provide a considerably more accurate modeling performance when compared to the independent category-wise GP as they can rely on the entirety of the training data rather than only the samples associated to a given category. This difference becomes more noticeable when the number of categories associated to the considered problem increases. This can, for instance, be seen when comparing the results obtained for the Branin and Goldstein functions in Figures 3.10 and 3.12, characterized respectively by 4 and 9 categories, but the same number of discrete variables. Among the considered kernel parameterizations, the CS is outperformed by the other kernels for most of the test-cases when considering sufficiently large data sets. This is due to the fact that the CS models the covariance between any pair of non identical levels with the same hyperparameter value, which considerably limits its modeling capabilities when confronted with complex problems and/or large number of levels. However, this difference in performance becomes less noticeable when small data sets are used, as the model is easier to train when compared to other kernels characterized by a larger number of hyperparameters. This can, for instance, be seen for the augmented Branin functions in Figure 3.11. The obtained results also show that, because of its specific distance-based construction, the LV kernel is not suitable when dealing with functions which present negative correlation trends between categories, as it can only return positive covariance values. This is for instance clearly shown in the results obtained for the Branin and augmented Branin functions in Figures 3.10 and 3.11. Finally, the hypersphere and coregionalization kernels show similar behaviors on the various considered test-cases, which can be explained by the similar construction and characteristics (*i.e.,* mapping of l levels onto an l-dimensional Hilbert space). The main differences in performance

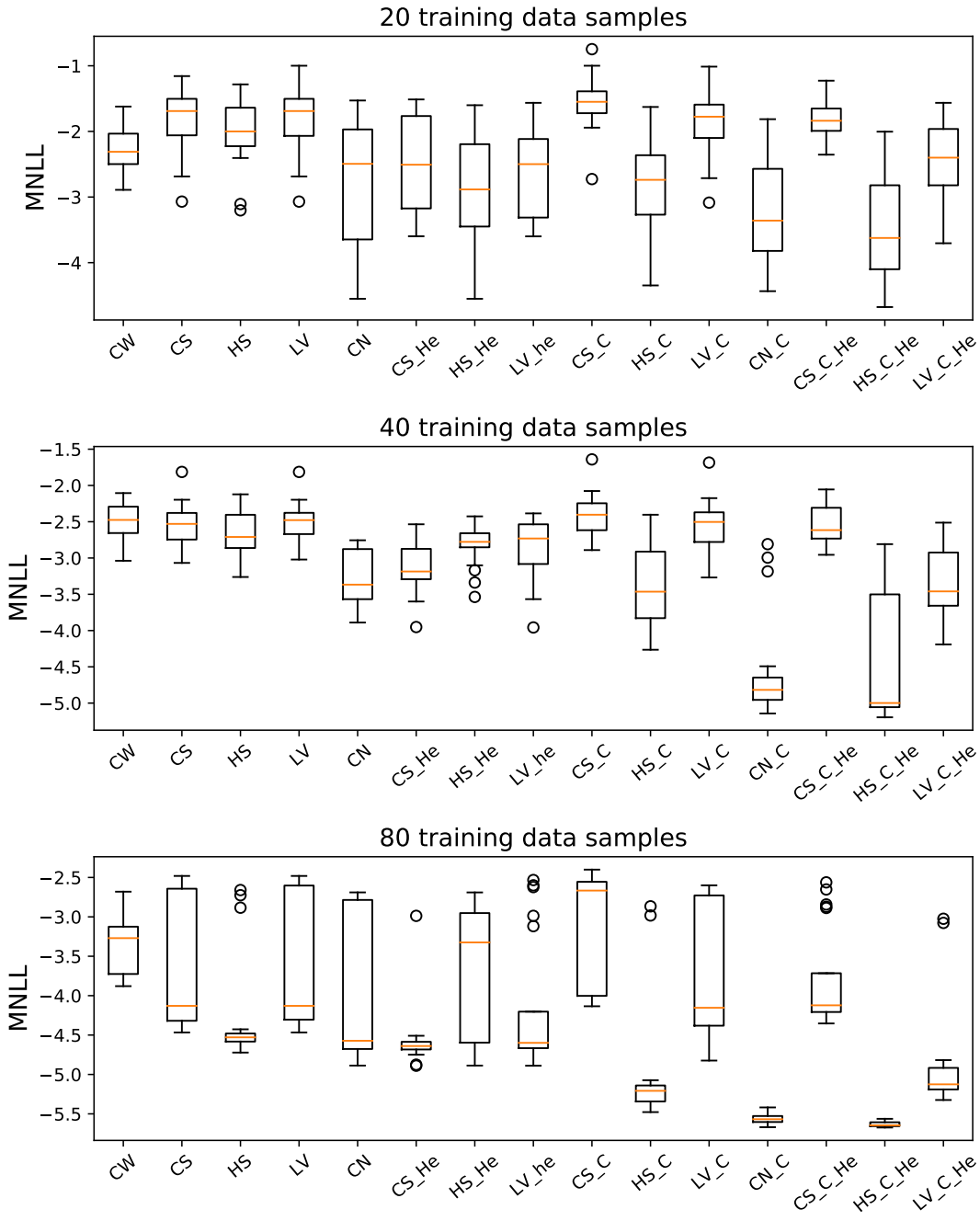


FIGURE 3.23: Comparison of various discrete kernels error model on the mixed-variable Branin function for various training data set sizes over 20 repetitions.

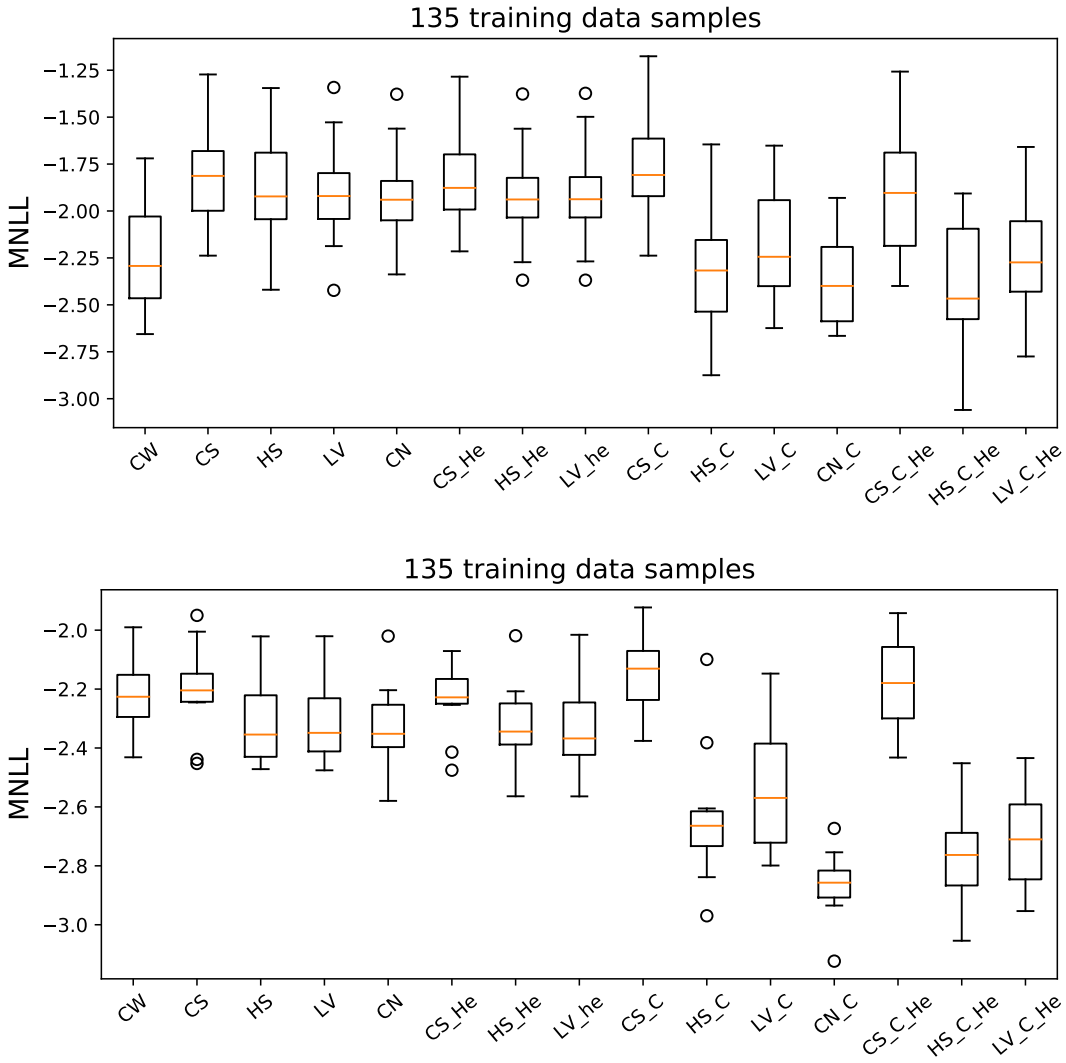


FIGURE 3.24: Comparison of various discrete kernels modeling MNLL on the thrust frame structural analysis test-case. The modeled values are the maximum inner skin Von Mises stress (top) and the upper interface longitudinal over-flux (bottom) over 10 repetitions.

between the two kernels can be identified when dealing with particularly small data sets, in which case the coregionalization is limited by its larger number of hyperparameters, and when dealing with heteroscedastic functions, which the coregionalization kernel can model inherently better.

The difference in performance between the level-wise and category-wise approach based kernels varies depending on the considered function. For mixed-variable functions characterized by a low number of categories (relatively to the number of discrete variables), a category-wise approach may provide a better modeling as it can allow to separately characterize the covariance between each category, thus better capturing the various trends of the considered function. This can for instance be seen in the results obtained for the Branin function in Figure 3.10. However, when considering functions characterized by larger number of categories, the level-wise approach based kernels tend to perform better than the category-wise based ones in case small training data sets are provided, as is for instance shown for the Goldstein function in Figure 3.12. This can be explained by the fact that the number of categories characterizing a given function tends to increase exponentially with the number of discrete variables. The number of hyperparameters necessary to characterize a category-wise kernel follows a similar trend and can therefore become considerably large with respect to the available data, thus resulting in a difficult model training process. For functions characterized by a particularly large number of categories, such as the fifth analytical benchmark, a category-wise approach becomes unfeasible, as it presents more categories than the amount of data samples which can be provided for the GP model training. An analysis of the considered function in terms of number of levels and categories with respect to the size of the available training data set might therefore be necessary in order to assess whether a level-wise or category-wise is more suitable.

The obtained results also show that in the presence of heteroscedastic trends in the modeled function, considering heteroscedastic kernels tends to results in better modeling performance, as can for instance be seen for the fifth analytical benchmark in Figure 3.15. However, it can also be noticed that when dealing with homoscedastic functions, considering heteroscedastic kernels usually yields results comparable with the homoscedastic ones. Therefore, unless the considered function presents a particularly large discrete design space or the training data is particularly limited, considering heteroscedastic kernels is usually the safest choice, unless problem specific knowledge is available.

Finally, the coherence of the considered discrete kernels error models is also tested on two representative test-cases (one analytical benchmark and one engineering design related benchmark) with different data set sizes by relying on the MNLL criterion. Overall, it is shown that the relative performance between kernels for a given modeled function is similar to the one obtained when considering the RMSE. The main noticeable difference with respect to the modeling accuracy benchmark is represented by the good performance of the independent CW GP modeling (*i.e.,* the reference method), especially for small sized training data sets, as is shown in Figures 3.23 and 3.24. In fact, in these cases the independent CW GP provides a more coherent error model than several of the compared kernels. This can be explained by the fact that this approach relies on considerably less data in order to build each one of the independent surrogate models, and provides therefore a more conservative variance prediction due to the lack of information with respect to a large portion of the search space.

3.10 Conclusions

In this chapter, the Gaussian Process based surrogate modeling of fixed-size mixed-variable functions is discussed. It is shown that it is possible to define a mixed-variable kernel by combining purely continuous and purely discrete kernels. Subsequently, the construction of valid discrete

kernels is addressed, and the existing alternatives are presented and compared. Furthermore, the resulting mixed-variable Gaussian processes are tested on a number of benchmarks with different characteristics. Overall, the obtained results show that relying on mixed-variable surrogate models rather than on separate and independent continuous GP for each category allows to better exploit the available data and by consequence model more accurately the considered functions. The results also show that depending on the specific characteristic of the modeled function, such as homoscedasticity, number of categories and presence of negative correlations, the relative performance of the compared kernel varies. As a result, the kernel choice must be adapted to the specifics of the considered problems.

Overall, in this chapter the modeling capabilities of mixed-variable Gaussian Processes when dealing with small training data sets are shown. This characteristic, coupled with the fact that GP can provide an estimate of the modeling error under the form of a variance as a (virtually) free bi-product of the modeled function prediction, makes mixed-variable GP a promising candidate for the surrogate model-based optimization of mixed-variable problems. For these reasons, the possibility of applying this technique within a Bayesian Optimization framework is explored in the following chapter.

CHAPTER 4

Mixed-variable Bayesian design optimization

4.1 Introduction

As is discussed in Chapter 2, the computational budget which can be used within the framework of a complex system design is usually limited due to the large computational cost of the objective and constraint functions characterizing the considered problem. In the context of this thesis, this translates into the necessity of converging towards the optimum neighborhood of mixed-variable problems while performing the lowest possible number of function evaluations. More specifically, in this chapter the fixed-size mixed-variable formulation presented in Eq.2.6 is considered. In this case, both the objective and constraint functions of the considered problem depend simultaneously on continuous and discrete design variables. For clarity purposes, the problem formulation is provided again below:

$$\begin{aligned} \min \quad & f(\mathbf{x},\mathbf{z}) \qquad f : F_x \times F_z \rightarrow F_f \subseteq \mathbb{R} \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x} \\ & \mathbf{z} \in F_z \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{z}) \leq 0 \\ & g_i : F_{x_i} \times F_{z_i} \rightarrow F_{g_i} \subseteq \mathbb{R} \qquad \text{for} \qquad i = 1,...,n_g \end{aligned} \tag{4.1}$$

Due to the computational cost of the considered problem functions, optimization algorithms commonly used in the presence of discrete variables such as mixed variable Genetic Algorithm [121] and Mesh Adaptive Discrete Search (MADS) [4], [5], are usually inadequate, as they require a large number of function evaluations in order to converge. Furthermore, they usually rely on a penalization-based handling of constraints, which tends to be inefficient if not properly tuned. An increasingly popular solution for computationally expensive problems is the Surrogate Model Based Design Optimization (SMBDO) [105], which involves surrogate models of the numerical functions characterizing the problem, created by using a data set of limited size, as already discussed in Chapter 3. These surrogate models are usually considerably cheaper to evaluate when compared to the exact problem functions, however, they also tend to introduce a modeling error which must be taken into account and dealt with. SMBDO consists in performing a simultaneous search for the problem optimum and refinement of the surrogate models by first determining the location in the design space at which the problem optimum is most likely to be found according to a given acquisition function (also referred to as infill criterion) and by then calculating the actual system performance and constraint values at this location. Subsequently, the newly computed data sample is added to the pre-existing data set and the surrogate models are updated. This routine is repeated until a given convergence criterion is reached. In the literature, several different SMBDO algorithms are proposed, each one relying on different Designs of Experiment (DoE),

different surrogate modeling techniques and different acquisition functions. Global overviews of the most popular SMBDO techniques and their various aspects can be found in [105]. More detailed reviews of surrogate modeling techniques for SMBDO purposes can be found in [41], [53], [117] and [128]. Finally, a review of the most popular data sample infill criteria is provided by Sasena [114].

Although an extensive literature exists on the use of different surrogate modeling techniques within the SMBDO framework, only a few surrogate modeling techniques for functions depending on both continuous and discrete variables exist in the literature [54], [62], [104], [108], [125], [134], [136], as is discussed in Chapter 3. Moreover, only a part of the aforementioned methods are actually developed and applied within an optimization framework, while the others are only developed for modeling purposes. A comprehensive review and taxonomy of the existing approaches and techniques allowing to perform mixed-variable SMBDO is provided Bartz-Beielstein and Zaefferer [16]. Among the mixed-variable surrogate modeling techniques applied for optimization purposes, one can find a few variants of Radial Basis Functions (RBF) based SMDBO techniques proposed in order to deal with mixed-integer (*i.e.,* continuous/discrete) problems [60], [73], [91], [106]. In these cases, no particular adaptation is required with respect to the continuous case, as the Euclidean distance-based working principle of the RBF can easily be applied to integer variables. In [90], [132] and [133] the possibility of performing mixed-variable SMBDO by relying on continuous surrogate modeling techniques, such as RBF, GP and linear models by replacing the use of Euclidean distance with a different definition is considered. More specifically, both the Hamming distance (sometimes referred to as Gower distance [51]) as well as swap and interchange distances based on the permutations between variable levels are considered. Similarly, in [54], [58] and [62], mixed-variable adaptations of the Efficient Global Optimization algorithm based on the use the Hamming distance are briefly described. A surrogate model assisted mixed-variable GA is discussed in [10] where both Generalized Linear Models (GLM) and RBF are considered. In this case, the discrete variables are handled by grouping the training data in clusters with respect to the discrete variable levels and subsequently defining a separate surrogate model for each cluster. The distance between clusters is then computed through the use of the Hamming distance. In [37] and [59] the effects of mapping the discrete variables onto dummy numerical variables (sometimes referred to as coding) in order to rely on continuous surrogate models is discussed. More specifically, [37] presents a mixed-variable adaptation of the moving least square regression scheme, while [59] discusses a mixed-variable multiple kernel regression-based Support Vector Machine (SVM) surrogate model. Both papers suggest that relying on the dummy coding of each discrete variable characterized by l levels onto a $l-1$-dimensional dummy binary variable tends to be the most convenient approach, thanks to the fact that it results in having all the levels equidistant from the others in the latent space, thus avoiding to insert a relation of order in the modeling of unordered variables. A nested Mixed-Integer Efficient Global Optimization Algorithm is proposed in [110] for the simultaneous design of an aircraft and the associated fleet allocation. This algorithm is then extended in [109] in order to solve a wing topology optimization problem. The proposed method is initialized with a given data set defined in the mixed continuous/integer design space. Each one of the samples is then used as a starting point for a gradient-based optimization within the continuous design space by fixing the integer variables. The obtained optimal values are subsequently used as a training data set for the creation of a GP model with respect to the integer variables. In order to reduce the number of hyperparameters to be trained, the GP model is combined with a Partial Least Square (PLS) dimension reduction approach [21]. This surrogate model is then used in order to define and optimize an acquisition function which is computed as a combination of the Expected Improvement (EI) and Expected Violation (EV) criteria, allowing to determine the most promising location of the integer design space at which to evaluate the actual problem functions. In order to perform the acquisition function optimization with respect to the integer variables a gradient-based B&B is proposed. Finally, an RBF assisted mixed-variable GA is discussed in [17]. Due to the proprietary nature

of the used software, no details on the actual implementation are provided.

Overall, the existing literature on mixed-variable SMBDO is relatively limited and, in most cases, focuses on either defining a concept of distance (*i.e.,* similarity or dissimilarity) applicable within the discrete or mixed-variable search space and/or on mapping the discrete variable levels onto numerical and ordered latent spaces. None of the methods discussed above propose a surrogate modeling technique which directly handles the mixed-variable nature of the design space. Moreover, most of existing mixed-variable SMBDO techniques are developed in order to deal with integer variables and cannot handle optimization problems depending on generic unordered discrete design variables. Finally, the handling of the constraints in the majority of the previously mentioned optimization methods relies on direct penalization of the objective function values for solutions that are not feasible. Although popular, this approach can be inadequate when confronted with expensive computations as it usually leads to a large number of function evaluations required in order for the considered optimization algorithm to converge.

In this chapter, a mixed-variable variant of the Bayesian Optimization (BO) algorithm, a SMBDO technique which relies on the use of GP surrogate models [107], is proposed. A very popular example of continuous BO is the Efficient Global Optimization (EGO) algorithm first proposed by Jones [63], which relies on the Expected Improvement (EI) acquisition function. As is discussed in Chapter 3, a few adaptations of GP allowing to model mixed-variable functions through the definition of discrete variable kernels have been proposed. However, none of the existing mixed-variable GP has been properly extended and applied within a BO framework, with the exception of recent works with the Latent Variable kernel [134]. The main contribution that is proposed in this chapter is a BO algorithm allowing to perform the SMBDO of constrained mixed variable problems by relying on the mixed continuous/discrete GP.

Following this introduction, a brief description of the working principles of BO is provided in the second section. Subsequently, the necessary steps required to adapt this particular type of optimization algorithm in order to solve mixed-variable optimization problems are discussed, with a focus on the extension of the acquisition function to the mixed-variable design space as well as its optimization. In the third section, the proposed mixed-variable BO algorithm is tested on a number of analytical and aerospace engineering related test-cases with different discrete kernel parameterizations. Finally, in Section 4 the obtained results are analyzed and the relevant conclusions are drawn. Please note that this chapter is an extension of the works presented in [96] and [99].

4.2 Mixed-variable Bayesian Optimization

Without loss of generality, BO can be decomposed into 2 main phases. The first phase, which is extensively discussed in Chapter 3, consists in creating a separate and independent GP based surrogate model of the objective function as well as each constraint by relying on a finite training data set. This chapter focuses on the second phase of BO, during which the most promising additional data samples, in terms of objective function value and feasibility, are identified, evaluated and added to the data set with the purpose of simultaneously refining the surrogate model (*i.e.,* improving the modeling accuracy and reducing its variance) and exploring the areas of the design space which are more likely to contain the optimization problem optimum. This refinement process is sometimes referred to as *infill*. The location at which each newly added data sample is computed is determined through an auxiliary optimization of a given acquisition function (or infill criterion). For clarity purposes, the BO algorithm is schematically represented in Figure 4.1.

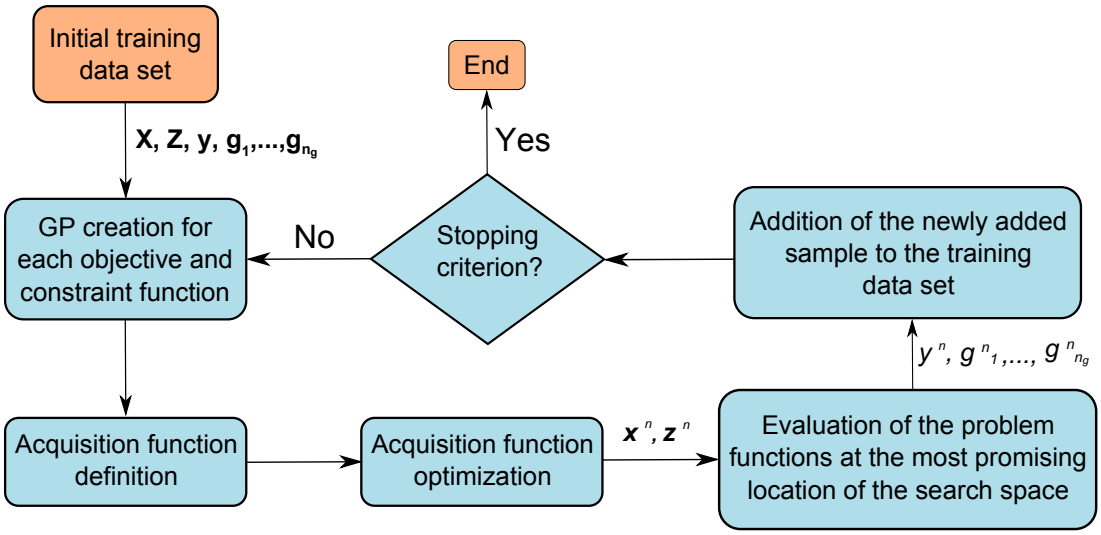


FIGURE 4.1: Schematic representation of the working principle of Bayesian Optimization. $\mathbf{X}$ and $\mathbf{Z}$ contain the continuous and discrete variable data sets, while $\mathbf{y}$ and $\mathbf{g}_1, \ldots, \mathbf{g}_{n_g}$ contain the associated objective function and constraint responses.

4.2.1 Mixed variable acquisition function

Depending on the characteristics of the considered problem, different acquisition functions may be considered. The common approach consists in defining this function as being comprised of 2 terms. The first one is tasked with determining the location of the design space at which the best value of the objective function is most likely to be found. The second term, instead, has the purposes of driving the infill search towards regions of the search space where the largest number of constraints is most likely to be satisfied.

4.2.1.1 Objective function oriented infill criterion

In this work, the first term of the acquisition function is defined as a mixed-variable adaptation of the Expected Improvement (EI) first defined in [89]. As the name suggests, the EI represents the expected value of the improvement $I = \max\left(y_{min} - Y(\mathbf{x}^*), 0\right)$ in terms of objective function value with respect to the data set. The original purely continuous formulation is the following:

$$\begin{aligned} \mathbb{E}[I(\mathbf{x}^*)] &= \mathbb{E}\left[\max\left(y_{min} - Y(\mathbf{x}^*), 0\right)\right] \\ &= (y_{min} - \hat{y}(\mathbf{x}^*))\Phi\left(\frac{y_{min} - \hat{y}(\mathbf{x}^*)}{\hat{s}(\mathbf{x}^*)}\right) + \hat{s}(\mathbf{x}^*)\phi\left(\frac{y_{min} - \hat{y}(\mathbf{x}^*)}{\hat{s}(\mathbf{x}^*)}\right) \end{aligned} \tag{4.2}$$

where y_{min} is the minimum feasible value present within the data set at the given BO iteration, $\hat{y}(\mathbf{x}^*)$ and $\hat{s}(\mathbf{x}^*)$ are the mean and standard deviation of the objective function prediction $Y(\mathbf{x}^*)$, $\Phi(.)$ is the cumulative distribution function of a normal distribution and finally $\phi(.)$ is the probability density function of a normal distribution. In practice, the EI provides a trade-off between the exploitation of the search space (*i.e.,* refinement of the incumbent solution), which is mostly driven by the first term of Eq. 4.3, and the exploration of the search space (*i.e.,* reduction of the modeling uncertainty over the entire design space), mostly driven by the second term.

The EI criterion is originally derived by considering purely continuous GP [89]. However, it can be easily shown that the derivation still holds in the mixed-variable case as long as the prediction provided by the GPs can be represented under the form of a normally distributed variable, *i.e.,* $\mathcal{N}\left(\hat{y}(\mathbf{x}^*, \mathbf{z}^*), \hat{s}^2(\mathbf{x}^*, \mathbf{z}^*)\right)$. As is discussed in Chapter 3, this property can be ensured by defining a valid mixed-variable kernel characterized by hyperparameters within their limit bounds. In the mixed-variable case, the EI criterion is defined over the mixed-variable design

space $F_x \times F_z$ and can then be defined as:

$$\begin{aligned} \mathbb{E}[I(\mathbf{x}^*, \mathbf{z}^*)] &= \mathbb{E}\left[\max\left(y_{min} - Y(\mathbf{x}^*, \mathbf{z}^*), 0\right)\right] \\ &= (y_{min} - \hat{y}(\mathbf{x}^*, \mathbf{z}^*))\Phi\left(\frac{y_{min} - \hat{y}(\mathbf{x}^*, \mathbf{z}^*)}{\hat{s}(\mathbf{x}^*, \mathbf{z}^*)}\right) + \hat{s}(\mathbf{x}^*, \mathbf{z}^*)\phi\left(\frac{y_{min} - \hat{y}(\mathbf{x}^*, \mathbf{z}^*)}{\hat{s}(\mathbf{x}^*, \mathbf{z}^*)}\right) \end{aligned} \tag{4.3}$$

For illustrative purposes, the following unconstrained continuous one-dimensional optimization problem is considered:

$$\begin{aligned} \min\ f(x) &:= \sin(x) + \sin\left(\frac{10}{3}x\right) \\ \text{w.r.t.} &\quad x \in [2.7, 7.5] \end{aligned} \tag{4.4}$$

The training data, the associated GP model and EI value over the entire design space over 5 iterations of the BO process are shown in Figure 4.2.

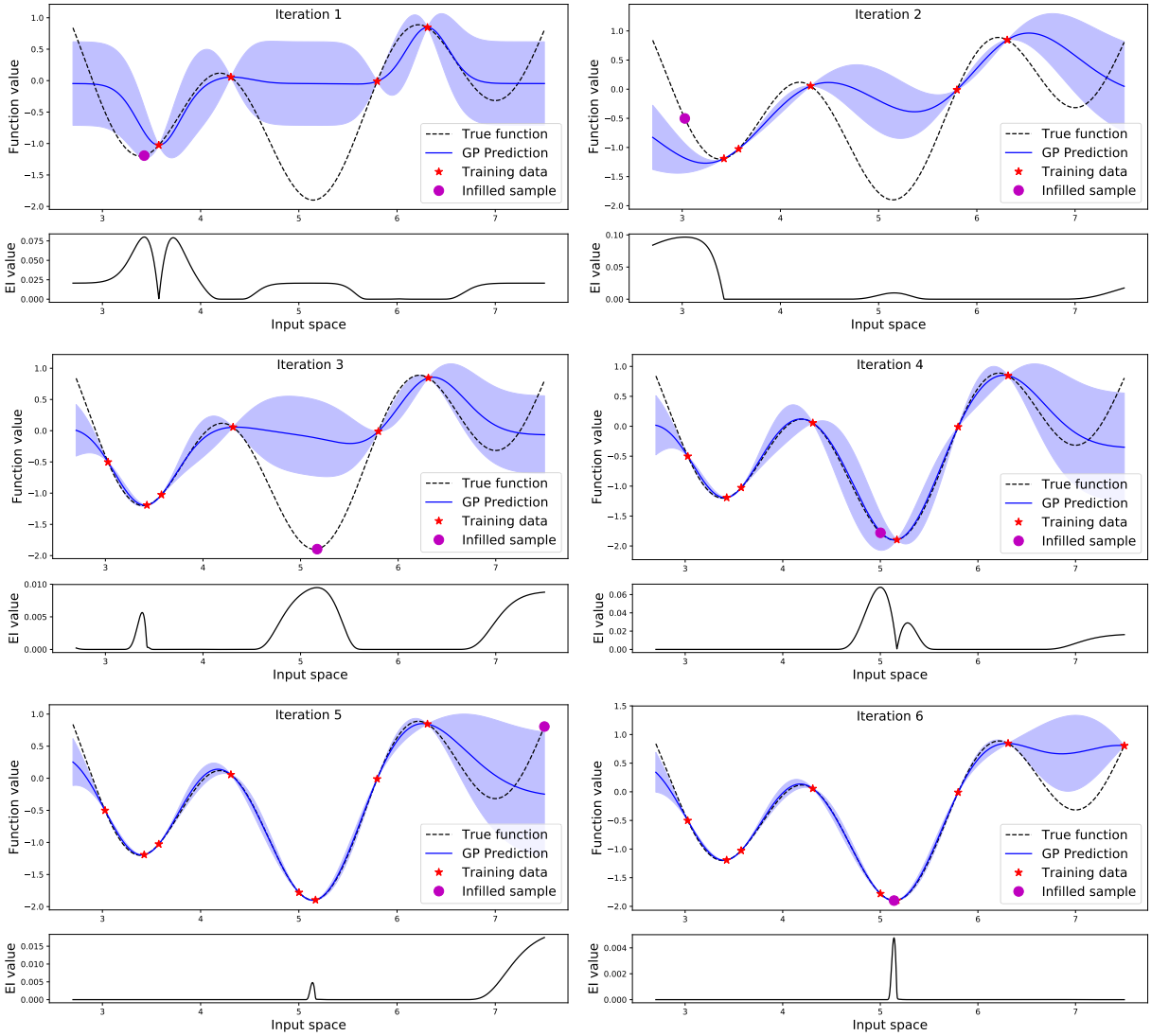


FIGURE 4.2: Example of Bayesian optimization of an unconstrained problem performed with the EI acquisition function over 6 iterations.

As is mentioned in the previous paragraph, the EI infill criterion is based on a trade-off between exploration and exploitation of the search space. Exploration allows to reduce the

modeling uncertainty over the most scarcely mapped areas of the design space, as happens for instance in iterations number 3 and 5 of the optimization presented in Figure 4.2. Exploitation, instead, has the objective of determining the location and objective function value of the problem optimum, and is for instance the purpose driving the infill location choice at the iteration number 4 of the considered example. It is also worth noticing that the maximum EI value tends to decrease along the optimization process, due to the reduction of the modeling uncertainty, and can be used in order to define the optimization stopping criterion, as is suggested in [63].

4.2.1.2 Feasibility oriented infill criteria

In order to take into account the presence of constraints and thus providing feasible final designs, the objective function oriented infill criterion discussed in the previous paragraph must be combined with an auxiliary criterion. Throughout this thesis, two alternative approaches are considered, however other solutions exist, as is for instance discussed in [33], [103], [114]. The first approach consists in computing the acquisition function as the product between the EI and the Probability of Feasibility (PoF) [116], which is defined as the probability that all the constraints the problem is subject to are satisfied at a given location of the search space $\{\mathbf{x}^*, \mathbf{z}^*\}$:

$$PoF(\mathbf{x}^*, \mathbf{z}^*) = \prod_{i=1}^{n_g} \mathbb{P}(G_i(\mathbf{x}^*, \mathbf{z}^*) \leq 0) \tag{4.5}$$

where $G_i(\mathbf{x}^*, \mathbf{z}^*)$ is the GP prediction of the constraint i at the considered unmapped location. Under the assumption that the GP prediction follows a normal distribution (*i.e.,* $G_i(\mathbf{x}^*, \mathbf{z}^*) \sim \mathcal{N}\left(\hat{g}_i(\mathbf{x}^*, \mathbf{z}^*), \hat{s}^2_{g_i}(\mathbf{x}^*, \mathbf{z}^*)\right)$), Eq. 4.5 can be rewritten as:

$$PoF(\mathbf{x}^*, \mathbf{z}^*) = \prod_{i=1}^{n_g} \Phi\left(\frac{0 - \hat{g}_i(\mathbf{x}^*, \mathbf{z}^*)}{\hat{s}_{g_i}(\mathbf{x}^*, \mathbf{z}^*)}\right) \tag{4.6}$$

Similarly to the EI, also the PoF definition can therefore be extended to the mixed-variable case in a fairly straightforward fashion, as long as the mixed-variable kernel is properly defined. The PoF can then be used in order to define an Infill Criterion (IC) allowing to optimize constrained problems as:

$$IC(\mathbf{x}^*, \mathbf{z}^*) = \mathbb{E}[I(\mathbf{x}^*, \mathbf{z}^*)] PoF(\mathbf{x}^*, \mathbf{z}^*) \tag{4.7}$$

The location $\{\mathbf{x}^n, \mathbf{z}^n\}$ at which the actual objective and constraint functions of the considered problem are evaluated at a given BO iteration can then be determined through the optimization of the following auxiliary problem:

$$\begin{aligned} \{\mathbf{x}^n, \mathbf{z}^n\} &= \operatorname{argmax}\left(IC(\mathbf{x}, \mathbf{z})\right) \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x} \\ & \mathbf{z} \in F_z \end{aligned} \tag{4.8}$$

Alternatively, the presence of constraints can be handled by relying on the Expected Violation (EV) criterion [8]. In a similar way to the EI, the EV represents the expected value of the violation of a given constraint, *i.e.,* the difference between the predicted value and the maximum acceptable value, which is usually set to 0:

$$V_i = \max\left(G_i(\mathbf{x}^*, \mathbf{z}^*) - 0, 0\right) \tag{4.9}$$

The EV for a given constraint $g_i(\cdot)$ is defined as follows:

$$\begin{aligned}\mathbb{E}[V_i(\mathbf{x}^*,\mathbf{z}^*)] &= \mathbb{E}[\max\left(G_i(\mathbf{x}^*,\mathbf{z}^*)-0,0\right)] \\ &= (\hat{g}_i(\mathbf{x}^*,\mathbf{z}^*)-0)\Phi\left(\frac{\hat{g}_i(\mathbf{x}^*,\mathbf{z}^*)-0}{\hat{s}_{g_i}(\mathbf{x}^*,\mathbf{z}^*)}\right)+\hat{s}_{g_i}(\mathbf{x}^*,\mathbf{z}^*)\phi\left(\frac{\hat{g}_i(\mathbf{x}^*,\mathbf{z}^*)}{\hat{s}_{g_i}(\mathbf{x}^*,\mathbf{z}^*)}\right)\end{aligned} \tag{4.10}$$

In the same way as the previously discussed infill criteria, the EV derivation still holds in the mixed-variable case as long as the GP prediction of each constraint follows a normal distribution (*i.e.,* $G_i(\mathbf{x}^*,\mathbf{z}^*) \sim \mathcal{N}\left(\hat{g}_i(\mathbf{x}^*,\mathbf{z}^*),\hat{s}^2_{g_i}(\mathbf{x}^*,\mathbf{z}^*)\right)$). If the EV is considered, the acquisition function is not defined as the product between infill criteria, but as a constrained problem, which has an impact on the choice of algorithm for the optimization of the acquisition function, as is discussed in the following paragraphs. In this case, the location $\{\mathbf{x}^n,\mathbf{z}^n\}$ at which the actual objective and constraint functions of the considered problem are evaluated at a given BO iteration is determined through the optimization of the following constrained auxiliary problem:

$$\begin{aligned}\{\mathbf{x}^n,\mathbf{z}^n\} &= \text{argmax}\left(EI(\mathbf{x},\mathbf{z})\right) \\ \text{s.t.} \quad & EV_i(\mathbf{x},\mathbf{z}) \leq t_i \quad \text{for} \quad i=1,\dots,n_g \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x \subseteq \mathbb{R}^{n_x} \\ & \mathbf{z} \in F_z\end{aligned} \tag{4.11}$$

where t_i is the maximum accepted violation for the constraint g_i. In other words, due to the fact that then original problem constraints (*i.e.,* $g_i(\cdot)$) are modeled under the form of random variables ($G_i(\cdot)$), data samples can be infilled at locations of the search space which are associated to non compliant expected constraint values, but only under a given tolerance threshold t_i. In general, lower values of t_i tend to drive the infill process towards local exploitation, whereas larger values enable a more widespread exploration of the design space.

Although both constraint oriented infill criteria are widely relied on, the PoF is more commonly used when dealing with problems characterized by a relatively low number of constraints thanks to its simplicity and to the fact that it results in an unconstrained acquisition function optimization problem. However, when dealing with problems with a large number of constraints, using the PoF often returns low acquisition function values over a large portion of the design space (as it is defined as a large product of values lower than 1). In other words, in these cases the PoF tends to outweigh the information provided by the EI, thus rendering the infill process more difficult [33]. Relying on the EV criterion usually allows to avoid this problem, however, it requires to perform a constrained acquisition function optimization. Furthermore, it also requires the user to determine a suitable violation threshold value for each considered constraint in order to provide efficient information to the BO process, which is a challenging task in case no information regarding the considered problem is known. The test-cases considered in the following paragraphs of this chapter present a fairly limited number of constraints. For this reason, the acquisition function considered in the remainder of this chapter is defined as the product between the EI and the PoF. In Chapter 5, instead, an EV-based acquisition function is considered in order to provide a fair comparison between candidate solutions subject to a different number of constraints.

As is schematically represented in Figure 4.1, once the values of $\{\mathbf{x}^*,\mathbf{z}^*\}$ that yield the optimal acquisition function value have been determined, the exact objective and constraint functions of the optimization problem can be computed at this location and the obtained data sample can be added to the GP training data set. Subsequently, the surrogate models must be trained anew in order to take into account the additional information provided by the newly computed data sample. This routine is repeated until a user-defined stopping criterion is reached. For instance,

in the original formulation of EGO the optimization is considered to have converged when the maximum EI value is smaller than 1% of the best current function value [63]. In order to better represent the conditions of real-life computationally intensive design problems, in this thesis a pre-defined number of data samples to be infilled is selected and the BO process is repeated until this computational budget is exhausted. This also allows to provide a fair comparison between the considered mixed-variable BO variants in terms of convergence speed.

Finally, it is important to mention that no assumption on the type of parameterization of the GP kernel is required when extending the infill criteria discussed above to the mixed-variable case. Therefore, the two possible acquisition functions proposed in this section are applicable with any of the discrete kernels described in Chapter 3.

4.2.1.3 Infill criterion optimization

For both acquisition functions defined in the previous paragraphs, an auxiliary optimization process is necessary in order to determine the location at which the newly infilled data sample is to be computed at each iteration of the BO algorithm. However, given that the computation time required to evaluate either acquisition functions is negligible when compared to the computation of the objective and constraint functions involved in complex system design problems, common optimization algorithms may be used. The acquisition function landscape is expected to present a large number of local optima in-between the locations of the training data-set samples. In the purely continuous case, two common approaches for the optimization of the acquisition function rely on either a gradient-based algorithm coupled with a large number of initializations or on global optimization algorithms, such as heuristic algorithms. However, the search space of the acquisition function optimization process corresponds to the design space of the considered problem. As a consequence, when dealing with mixed-variable problems the acquisition function must be optimized within the mixed continuous/discrete search space and is therefore subject to part of the limitations discussed in Chapter 2 related to the presence of discrete and unoredered variables. In this case, the acquisition function optimization can either be directly performed in the mixed continuous/discrete search space or separately in each category of the considered problem by subsequently choosing the category yielding the largest acquisition function value. The latter approach tends to become computationally inefficient, if not unfeasible, when dealing with mixed-variable problems characterized by large number of categories. For this reason, the results presented in this work are obtained with the first approach, by optimizing the infill criterion with the help of a mixed continuous/discrete Genetic Algorithm (GA) similar to the one presented by Stelmack [121] and coded with the help of the python based toolbox DEAP [43]. More specifically, an unconstrained GA is considered when relying on a PoF based acquisition function. If the constraints are instead handled through the use of the EV criterion, the acquisition function is optimized with the help of a GA with constraint dominance selection criterion [126]. The specific parameters of this mixed-variable GA, such as mutation and cross-over probabilities as well as number of generations and population size, vary between test-cases and are adapted to the size and characteristics of the considered problem.

4.3 Applications and Results

In order to assess the actual performance of the proposed mixed-variable BO algorithm, it is applied to a number of test-cases. Part of these benchmarks are related to the ones presented in Chapter 3, thus facilitating to draw connections between the results obtained within a modeling framework and the ones obtained within an optimization framework. For each benchmark, the algorithm is tested by relying on different discrete kernels described in Chapter 3. More specifically, the following kernels are considered: Compound Symmetry (CS), Latent Variable (LV), Hypersphere decomposition (HS) and Coregionalization (CN). Furthermore, homoscedastic and

heteroscedastic (_He) variants are considered, as well as level-wise and category-wise (_C) kernel definitions. The objective of this benchmark is to assess the impact of the problem characteristics as well as the parameterization of the considered discrete kernel choices on the convergence speed of the proposed BO method. In order to provide a measure of comparison, the results obtained with the proposed variants of the mixed-variable optimization algorithm are compared with a penalized mixed variable GA similar to the one proposed in [121], as well as a BO algorithm based on separate and independent GP for each category of the problem (referred to with the acronym CW). In other words, this last method relies on the independent continuous category-wise GP which are used as a reference in Chapter 3. A separate acquisition function is then defined and separately optimized in each category and the data sample to be infilled at a given iteration is computed as the location yielding the largest acquisition function value among every category.

The initial data set which is provided to the BO algorithm is sampled in the same way as in Chapter 3, by evenly and randomly distributing the data obtained through a single continuous LHS [86] between all of the problem categories. Furthermore, in order to quantify and compensate the influence of the initial DoE random nature, each optimization problem is solved multiple times with different initial training data sets. The actual number of repetitions depends on the optimization problem which is being considered.

4.3.1 Benchmark analysis

In the following paragraphs, the proposed mixed-variable BO is tested on a number of test-cases with different kernel parameterizations. More specifically, 3 analytical functions and 2 engineering-related test-cases are considered. These benchmarks present different characteristics in terms of continuous and discrete design space dimensions, combinatorial design space size, complexity, presence of negative correlation and heteroscedastic trends and category-wise construction. The main properties of the considered test-cases as well as the simulation details are provided below:

Constrained Branin function

- 2 continuous dimensions, 2 discrete dimensions, 9 categories
- 1 constraint
- Initial data set size: 12 samples
- Number of infilled samples: 20
- Compared methods: CW, CS, CN, HS_He, CN_C, HS_C_He, GA
- Acquisition function: $EI * PoF$
- Simple low-dimension mixed-variable test-case

Augmented Branin function

- 10 continuous dimensions, 2 discrete dimensions, 4 categories
- 1 constraint
- Initial data set size: 40 samples
- Number of infilled samples: 120
- Compared methods: CW, CS, CN, HS_He, CN_C, HS_C_He, GA
- Acquisition function: $EI * PoF$

- Increase of the continuous design space size with respect to the Branin function (but identical discrete design space characteristics).

Goldstein function

- 2 continuous dimensions, 2 discrete dimensions, 9 categories
- 1 constraint
- Initial data set size: 27 samples
- Number of infilled samples: 30
- Compared methods: CW, CS, LV, CN, HS_He, CN_C, HS_C_He, GA
- Acquisition function: $EI * PoF$
- Increase in the number of categories with respect to the Branin function (but identical discrete design space size).

Launch vehicle propulsion performance optimization

- 4 continuous dimensions, 3 discrete dimensions, 24 categories
- 8 constraints
- Initial data set size: 72 samples
- Number of infilled samples: 150
- Compared methods: CS, LV, CN, HS_He
- Acquisition function: $EI * PoF$
- Complex and realistic simulation. Larger number of categories as well as constraints compared to the analytical test-cases.

Thrust frame structural optimization

- 12 continuous dimensions, 2 discrete dimensions, 9 categories
- 5 constraints
- Initial data set size: 45 samples
- Number of infilled samples: 150
- Compared methods: CW, CS, LV, CN, HS_He
- Acquisition function: $EI * PoF$
- Realistic simulation. Linear trends with respect to the continuous sizing variables for the objective function.

Implementation

Similarly to the modeling performance benchmark analysis of Chapter 3, the results presented in the following paragraphs are obtained with the following implementation. The optimization routine overhead is written in Python 3.6. The GP models are created with the help of GPflow [83], a Python based toolbox for GP-based modeling relying on the Tensorflow framework [1] (version 1.13). The surrogate model training is performed with the help of a Bounded Limited memory Broyden - Fletcher - Goldfarb – Shanno (L-BFGS-B) algorithm [24], whereas the acquisition functions are optimized the help of a mixed continuous/discrete Genetic Algorithm (GA) [121] implemented by relying on the Python based toolbox DEAP [43].

4.3.2 Branin function

The first analytical benchmark to be considered is a constrained version of the mixed-variable Branin function presented in Chapter 3. It depends on 2 continuous variables and 2 discrete variables, each one presenting 2 levels, thus resulting in a total of 4 discrete categories. This optimization problem is defined as follows:

$$\begin{aligned} &\min f(x_1, x_2, z_1, z_2) \\ &\text{w.r.t.} \quad x_1, x_2, z_1, z_2 \end{aligned} \tag{4.12}$$

$$\text{s.t.} \quad g(x_1, x_2, z_1, z_2) \geq 0 \tag{4.13}$$

with:

$$x_1 \in [0,1], \quad x_2 \in [0,1], \quad z_1 \in \{0,1\}, \quad z_2 \in \{0,1\}$$

where:

$$f(x_1, x_2, z_1, z_2) = \begin{cases} h(x_1, x_2) & \text{if } z_1 = 0 \text{ and } z_2 = 0 \\ 0.4h(x_1, x_2) & \text{if } z_1 = 0 \text{ and } z_2 = 1 \\ -0.75h(x_1, x_2) + 3.0 & \text{if } z_1 = 1 \text{ and } z_2 = 0 \\ -0.5h(x_1, x_2) + 1.4 & \text{if } z_1 = 1 \text{ and } z_2 = 1 \end{cases} \tag{4.14}$$

$$\begin{aligned} h(x_1, x_2) = \Bigg[\Bigg(&(15x_2 - \frac{5}{4\pi^2}(15x_1 - 5)^2 + \frac{5}{\pi}(15x_1 - 5) - 6)^2 + \\ &10\left(1 - \frac{1}{8\pi}\right)\cos(15x_1 - 5) + 10\Bigg) - 54.8104\Bigg] \frac{1}{51.9496} \end{aligned} \tag{4.15}$$

$$g(x_1, x_2, z_1, z_2) = \begin{cases} x_1 x_2 - 0.4 & \text{if } z_1 = 0 \text{ and } z_2 = 0 \\ 1.5x_1 x_2 - 0.4 & \text{if } z_1 = 0 \text{ and } z_2 = 1 \\ 1.5x_1 x_2 - 0.2 & \text{if } z_1 = 1 \text{ and } z_2 = 0 \\ 1.2x_1 x_2 - 0.3 & \text{if } z_1 = 1 \text{ and } z_2 = 1 \end{cases} \tag{4.16}$$

The compared BO techniques are initialized with a training data set containing 12 samples (*i.e.*, 3 samples for each continuous independent model in the CW case) and subsequently 20 additional data points are infilled during the optimization process. Please note that as both discrete variables are characterized by 2 levels, the LV kernel is not considered, due to the fact that it would be equivalent to the CS one, as is shown in Chapter 3. The results obtained for the optimization of this constrained Branin function over 20 repetitions are presented in Figure 4.3. As a reference, the average number of function evaluations required by a penalized mixed-variable GA (the same one used to optimize the infill criterion) in order to reach the same median optimum value as the compared mixed-variable BO algorithms over 20 repetitions is larger than 800 (compared to the 32 provided to the BO methods), as is shown in Figure 4.4.

The obtained results show that for this first analytical benchmark, the proposed mixed-variable BO algorithm converges faster than the standard independent category-wise approach as well as the GA. Indeed, all the proposed mixed-variable BO algorithm variants consistently converge to the problem optimum, whereas the independent category-wise approach is not provided with enough computational budget in order to be able to do the same. Similarly, the GA algorithm requires a vastly larger number of function evaluations in order to converge. Among the proposed mixed-variable methods, no considerable difference in convergence rate can be noticed, with the exception of the LV kernel which requires a larger number of infilled data points in order

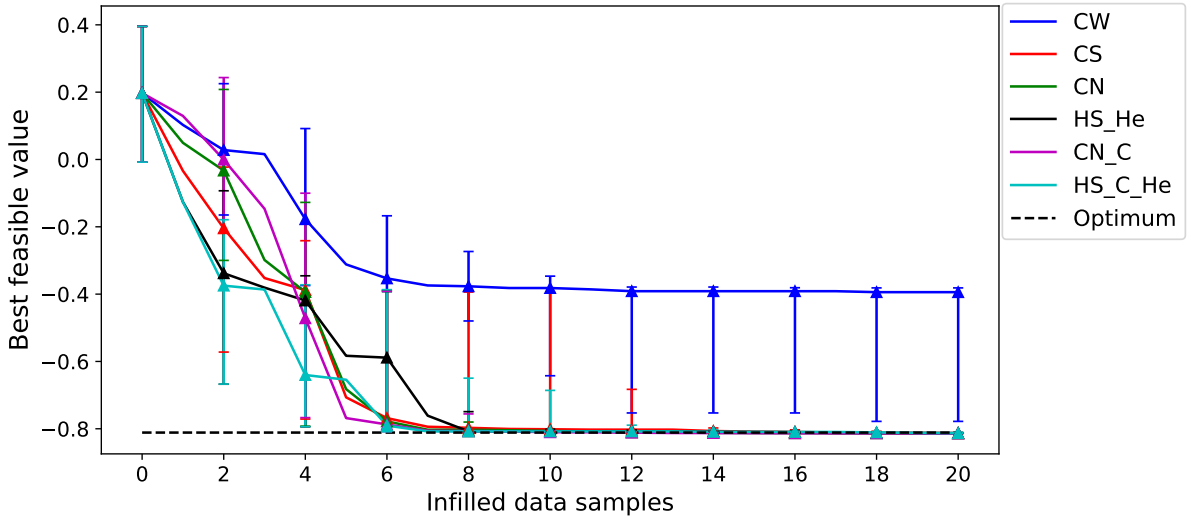


FIGURE 4.3: Comparison of the convergence rate of various discrete kernels during the BO of the mixed-variable Branin function over 20 repetitions.

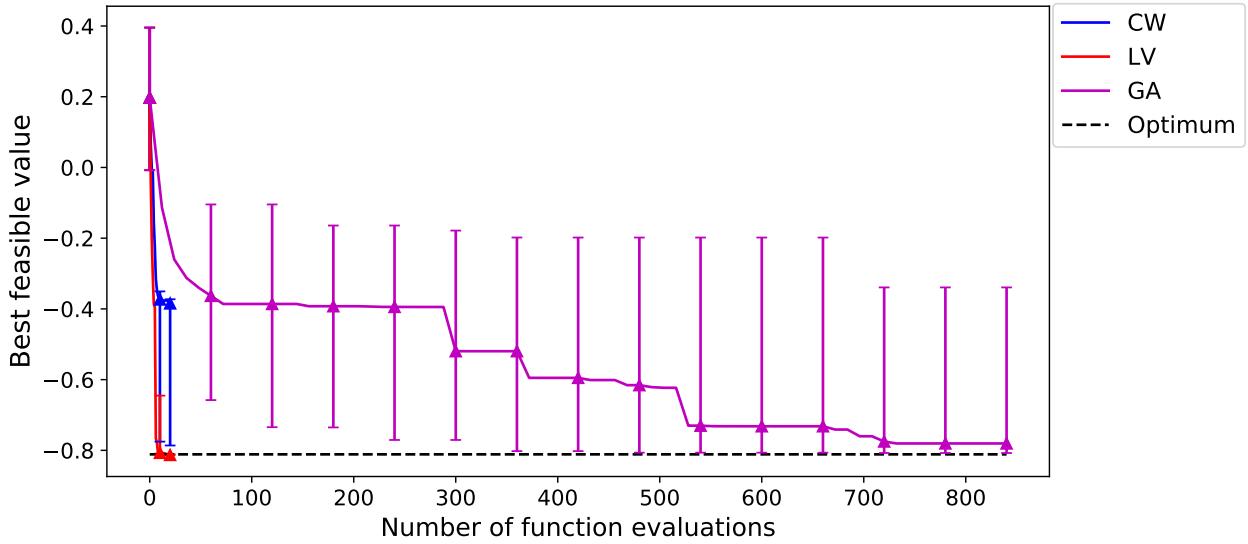


FIGURE 4.4: Comparison of the convergence rates of the proposed BO algorithms and a penalized mixed-variable GA for the mixed-variable Branin function over 20 repetitions.

to reduce the variance of the convergence rate over the various repetitions. As is shown in Chapter 3, this can be explained by the fact that the LV kernel is less performant when dealing with modeled functions which present negative correlation trends between levels or categories, as it can only return positive covariance values. It is also interesting to note that no significant difference in performance can be noted between the level-wise and category-wise modeling approaches for the considered kernel parameterizations (*e.g.,* HS_He and HS_C _He).

For illustrative purposes, the locations of the data samples infilled along the optimization process by the different considered methods during one of the repetitions are presented in Figure 4.5. Overall, it can be seen that the 3 considered mixed-variable BO are able to easily identify the category and the neighborhood of the global optimum. As a consequence, a large number of data samples are infilled within this neighborhood, with the exception of a few 'exploration' points on the feasibility threshold and on the bounds of the design space. The CW independent

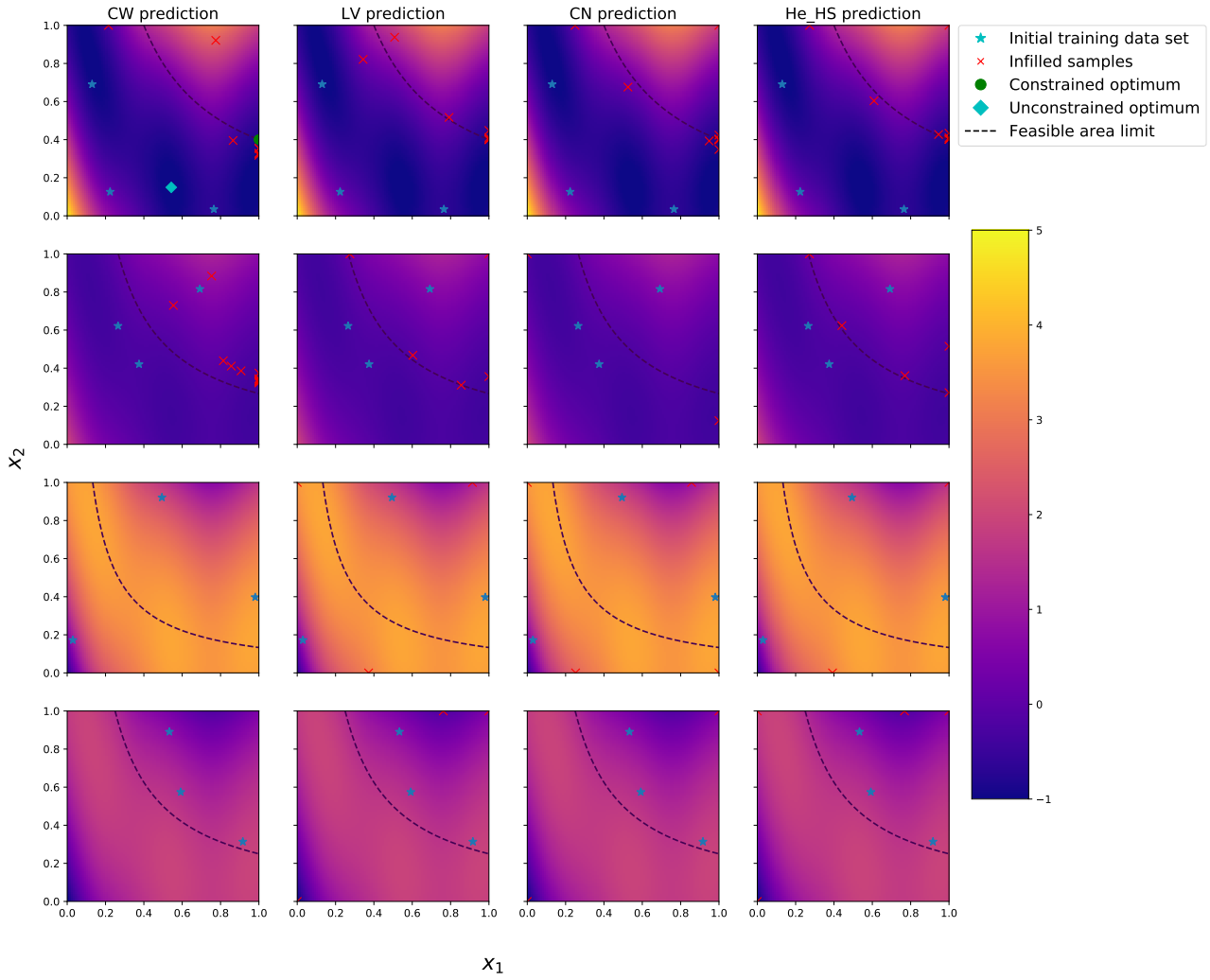


FIGURE 4.5: Location of the infilled data samples during one repetition of the Branin test-case optimization for various kernel parameterizations.

BO, instead, suffers from its lower modeling accuracy and needs to explore the mixed variable search space before being able to identify the correct category as well as the area of interest. By consequence, a larger amount of function evaluations are performed in the non-optimal categories and areas of the problem. Most notably, it can be seen that the CW independent BO infills a large number of data samples in both the first and second category, as it does not seem able to identify the optimal one (*i.e.,* the first one).

4.3.3 Augmented Branin function

The second analytical optimization problem which is considered in this thesis is the augmented augmented Branin function, introduced in Chapter 3, with the inclusion of a constraint. It is characterized by 10 continuous variables and 2 discrete variables, for a total of 4 categories and is defined as follows:

$$\min f(x_1, ..., x_{10}, z_1, z_2) \tag{4.17}$$
$$\text{w.r.t.} \quad x_1, \dots, x_{10}, z_1, z_2$$

$$\text{s.t.} \quad g(x_1, ..., x_{10}, z_1, z_2) \geq 0 \tag{4.18}$$

with:

$$x_1, ..., x_{10} \in [0,1], \quad z_1 \in \{0,1\}, \quad z_2 \in \{0,1\}$$

where:

$$f(x_1, ..., x_{10}, z_1, z_2) = \begin{cases} \tilde{h}(x_1, ..., x_{10}) & \text{if } z_1 = 0 \text{ and } z_2 = 0 \\ 0.4\tilde{h}(x_1, ..., x_{10}) + 1.1 & \text{if } z_1 = 0 \text{ and } z_2 = 1 \\ -0.75\tilde{h}(x_1, ..., x_{10}) + 5.2 & \text{if } z_1 = 1 \text{ and } z_2 = 0 \\ -0.5\tilde{h}(x_1, ..., x_{10}) - 2.1 & \text{if } z_1 = 1 \text{ and } z_2 = 1 \end{cases} \tag{4.19}$$

and:

$$\tilde{h}(x_1, ..., x_{10}) = \frac{h(x_1, x_2) + h(x_3, x_4) + h(x_5, x_6) + h(x_7, x_8) + h(x_9, x_{10})}{5} \tag{4.20}$$

with:

$$\begin{aligned} h(x_i, x_j) = &(((15x_j - \frac{5}{4\pi^2}(15x_i - 5)^2 + \frac{5}{\pi}(15x_i - 5) - 6)^2 + \\ &10\left(1 - \frac{1}{8\pi}\right)\cos(15x_i - 5) + 10) - 54.8104)/51.9496 \end{aligned} \tag{4.21}$$

$$g(x_1, ..., x_{10}, z_1, z_2) = \begin{cases} \tilde{g}(x_1, ..., x_{10}) - 0.3 & \text{if } z_1 = 0 \text{ and } z_2 = 0 \\ 0.4\tilde{g}(x_1, ..., x_{10}) - 0.4 & \text{if } z_1 = 0 \text{ and } z_2 = 1 \\ -0.75\tilde{g}(x_1, ..., x_{10}) - 0.2 & \text{if } z_1 = 1 \text{ and } z_2 = 0 \\ -0.5\tilde{g}(x_1, ..., x_{10}) - 0.3 & \text{if } z_1 = 1 \text{ and } z_2 = 1 \end{cases} \tag{4.22}$$

$$\tilde{g}(x_1, ..., x_{10}) = \sum_{i=1}^{5} x_{2i} \cdot x_{2i+1} \tag{4.23}$$

For the compared BO techniques, an initial training data set of 40 samples is used (*i.e.,* 10 samples for each independent GP in the CW case) and subsequently 120 additional data samples are infilled during the BO process. Similarly to the previous test-case, the LV kernel is not considered, as it would be equivalent to the CS one due to the presence of 2 levels for each considered discrete variable. The results obtained for the optimization of the augmented Branin function averaged over 10 repetitions are shown in Figures 4.6 and 4.7. As a reference, the average number of function evaluations required by the penalized mixed-variable GA in order to reach the same median optimum value as the considered mixed-variable BO algorithms over 20 repetitions is equal to 2500 (compared to the 160 provided to the BO methods). Similarly to the previous test-case, the proposed mixed-variable BO adaptations provide more robust results when few function evaluations are allowed if compared to the reference CW BO method. Furthermore, the results also show that a slightly better performance from both the category-wise and level-wise variants of the coregionalization kernel. However, differently than with the previous Branin function, it can also be noted that over the 20 repetitions none of the compared methods converges to the problem global optimum. Nevertheless, an analysis of the results shows that all the compared kernels are consistently able to identify the optimal category of the problem (*i.e.,* the incumbent optimum at the end of each mixed-variable BO is located in said category), and it can therefore be stated that converging to the global optimum becomes a matter of refining the incumbent solution in the continuous design space.

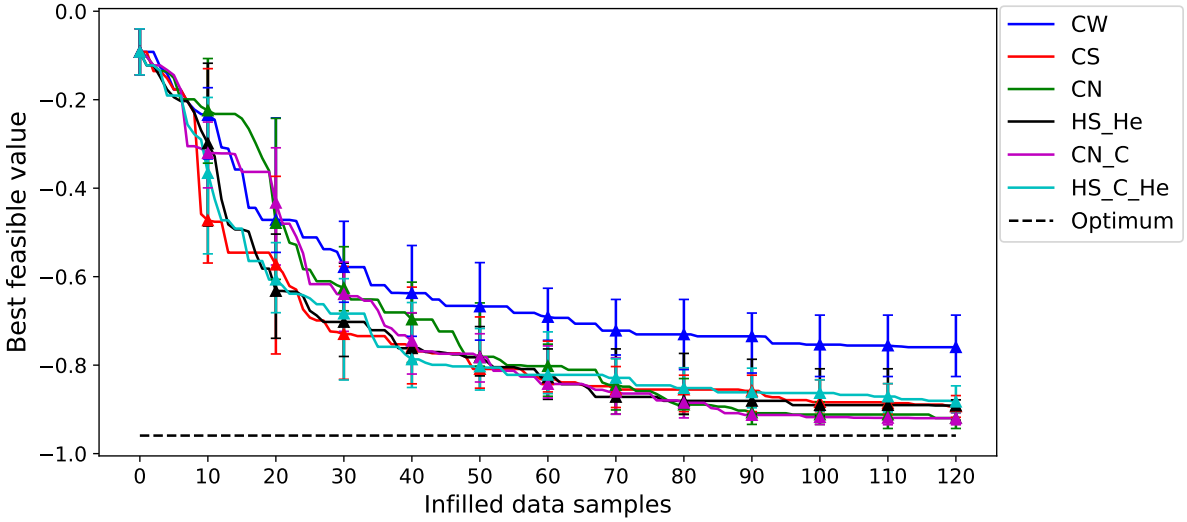


FIGURE 4.6: Comparison of the convergence rate of various discrete kernels during the BO of the augmented mixed-variable Branin function over 20 repetitions.

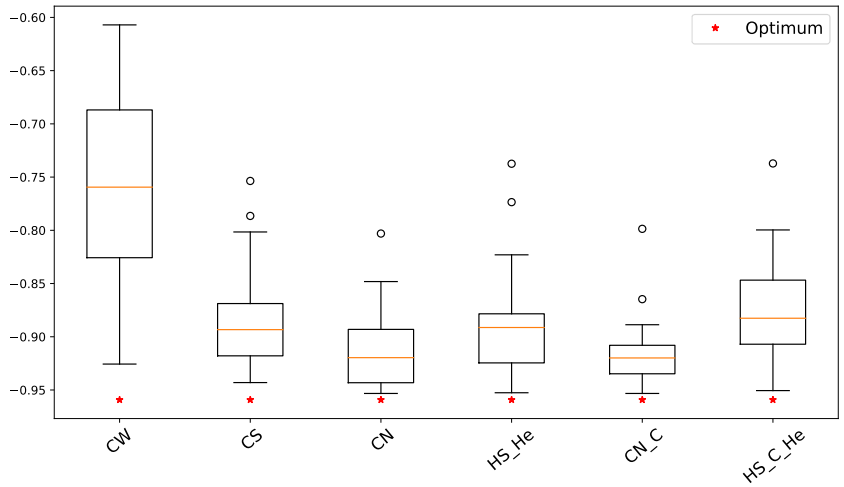


FIGURE 4.7: Comparison of the convergence values of various discrete kernels during the BO of the augmented mixed-variable Branin function over 20 repetitions.

4.3.4 Goldstein function

The third and last analytical benchmark function to be considered is a constrained variant of the mixed-variable Goldstein function described in Chapter 3 characterized by 2 continuous variables, 2 discrete variables and 9 discrete categories. This optimization problem is defined as follows:

$$\begin{aligned} &\min f(x_1, x_2, z_1, z_2) \\ &\text{w.r.t.} \quad x_1, x_2, z_1, z_2 \end{aligned} \tag{4.24}$$

$$\text{s.t.} \quad g(x_1, x_2, z_1, z_2) \geq 0 \tag{4.25}$$

with:

$$x_1 \in [0, 100], \quad x_2 \in [0, 100], \quad z_1 \in \{0, 1, 2\}, \quad z_2 \in \{0, 1, 2\}$$

where:

$$f(x_1, x_2, z_1, z_2) = h(x_1, x_2, x_3, x_4) \tag{4.26}$$

$$\begin{aligned} h(x_1, x_2, x_3, x_4) = \; & 53.3108 + 0.184901 x_1 - 5.02914 x_1^3 \cdot 10^{-6} + 7.72522 x_1^4 \cdot 10^{-8} - \\ & 0.0870775 x_2 - 0.106959 x_3 + 7.98772 x_3^3 \cdot 10^{-6} + \\ & 0.00242482 x_4 + 1.32851 x_4^3 \cdot 10^{-6} - 0.00146393 x_1 x_2 - \\ & 0.00301588 x_1 x_3 - 0.00272291 x_1 x_4 + 0.0017004 x_2 x_3 + \\ & 0.0038428 x_2 x_4 - 0.000198969 x_3 x_4 + 1.86025 x_1 x_2 x_3 \cdot 10^{-5} - \\ & 1.88719 x_1 x_2 x_4 \cdot 10^{-6} + 2.50923 x_1 x_3 x_4 \cdot 10^{-5} - \\ & 5.62199 x_2 x_3 x_4 \cdot 10^{-5} \end{aligned} \tag{4.27}$$

$$g(x_1, x_2, z_1, z_2) = c_1 \sin\left(\frac{x_1}{10}\right)^3 + c_2 \cos\left(\frac{x_2}{20}\right)^2 \tag{4.28}$$

The values of x_3, x_4, c_1 and c_2 are determined as a function of z_1 and z_2 according to the relations provided in Table 4.1. For the SMBDO techniques, an initial training data set of 27

	$z_1 = 0$		$z_1 = 1$		$z_1 = 2$	
$z_2 = 0$	$x_3 = 20$	$x_4 = 20$	$x_3 = 50$	$x_4 = 20$	$x_3 = 80$	$x_4 = 20$
	$c_1 = 2$	$c_2 = 0.5$	$c_1 = -2$	$c_2 = 0.5$	$c_1 = 1$	$c_2 = 0.5$
$z_2 = 1$	$x_3 = 20$	$x_4 = 50$	$x_3 = 50$	$x_4 = 50$	$x_3 = 80$	$x_4 = 50$
	$c_1 = 2$	$c_2 = -1$	$c_1 = -2$	$c_2 = -1$	$c_1 = 1$	$c_2 = -1$
$z_2 = 2$	$x_3 = 20$	$x_4 = 80$	$x_3 = 50$	$x_4 = 80$	$x_3 = 80$	$x_4 = 80$
	$c_1 = 2$	$c_2 = -2$	$c_1 = -2$	$c_2 = -2$	$c_1 = 1$	$c_2 = -2$

TABLE 4.1: Characterization of the Goldstein function discrete categories

samples is used (*i.e.,* 3 samples for each independent EGO in the CW case) and subsequently 30 additional data points are infilled during the optimization process. The results obtained for the optimization of the constrained Goldstein function over 20 repetitions are shown in Figures 4.8 and 4.10. As a reference, the average number of function evaluations required by a penalized mixed-variable GA (the same one used to optimize the infill criterion) in order to reach the same median optimum value as the compared mixed-variable BO algorithms over 20 repetitions is equal to 1600 (compared to the 57 provided to the BO methods). Also in this case, the difference in

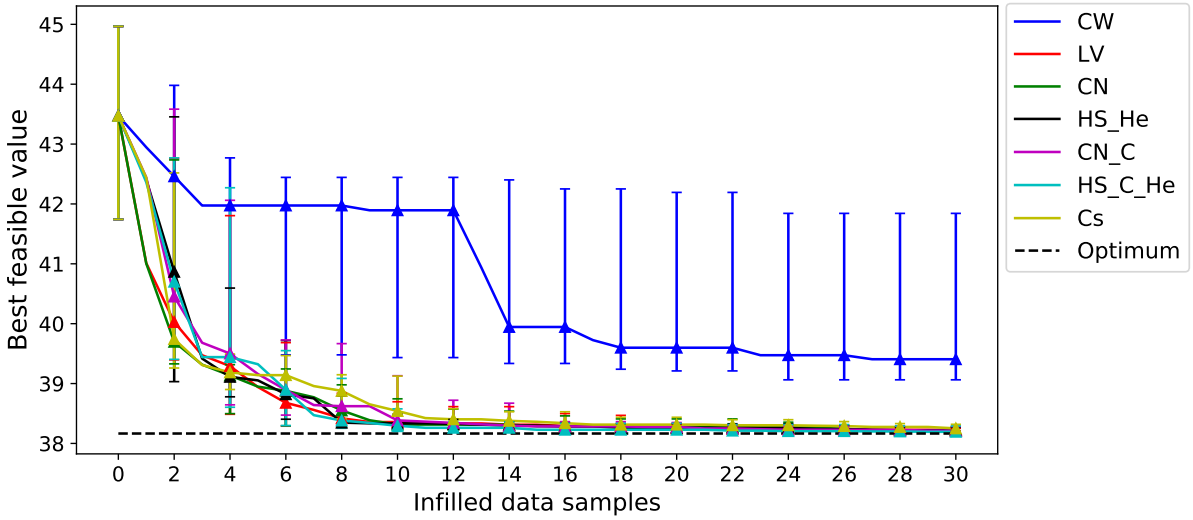


FIGURE 4.8: Comparison of the convergence rate of various discrete kernels during the BO of the mixed-variable Goldstein function over 20 repetitions.

performance between the independent CW modeling based BO and the proposed mixed-variable adaptations is considerable. In fact, it can be seen that these adaptations provide a consistent convergence towards the neighborhood of the problem optimum, which is instead never the case for the CW BO, as it is not provided with sufficient function evaluations. Overall, no noticeable difference in performance between the various considered discrete kernels can be seen, in terms of convergence speed. However, the objective function value at convergence shows a slightly better and more consistent performance from the category-wise heteroscedastic HS kernel, and a slightly worse performance from the CS kernel, which is consistent with the modeling results obtained in Chapter 3.

4.3.5 Launch vehicle propulsion performance optimization

In order to show the potential applications of the presented mixed variable BO techniques for actual engineering problems, the design optimization of a solid propulsion engine for a sounding rocket is considered. Sounding rockets carry scientific experiments into space along a parabolic trajectory. Their overall time in space is brief and the cost factor makes them an interesting alternative to heavier launch vehicles as they are sometimes more appropriate to successfully carry out a scientific mission and are less complex to design. The objective of the considered multidisciplinary design optimization problem is to maximize the speed increment (ΔV) of a solid propulsion sounding rocker under geometrical, propulsion and structural constraints. Three disciplines are involved in the considered test case: the propulsion, the mass budget and geometry design and the structural sizing, as is shown in Figure 4.11. This mixed variable optimization is characterized by a total of 24 categories. Due to the large number of categories, the CW independent GP based BO is not considered for this test-case. The various continuous and discrete design variables characterizing its performance and constraint functions are detailed in Table 4.2. The equations provided below can be found in [70]. The objective function, which is defined as the speed increment ΔV, can be computed through the so-called Tsiolkovsky rocket equation:

$$\Delta V = g_0 I_{sp} \ln\left(\frac{M_i}{M_f}\right) \tag{4.29}$$

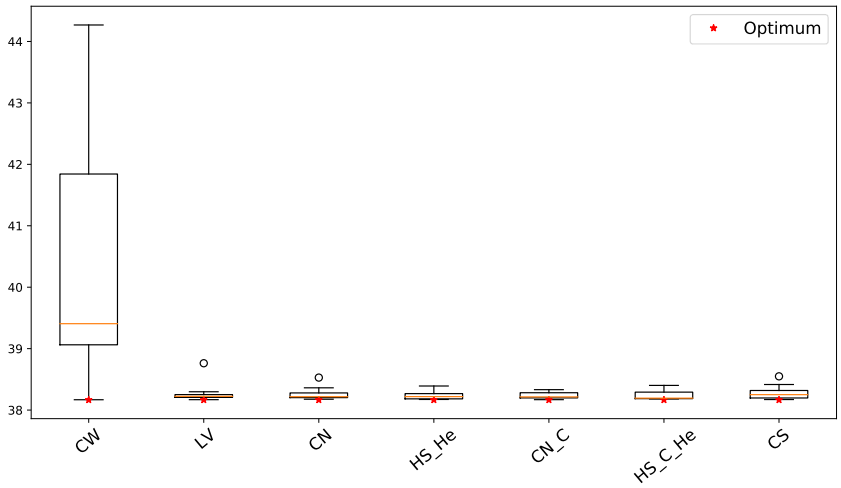


FIGURE 4.9: Comparison of the convergence value of various discrete kernels during the BO of the mixed-variable Goldstein function over 20 repetitions.

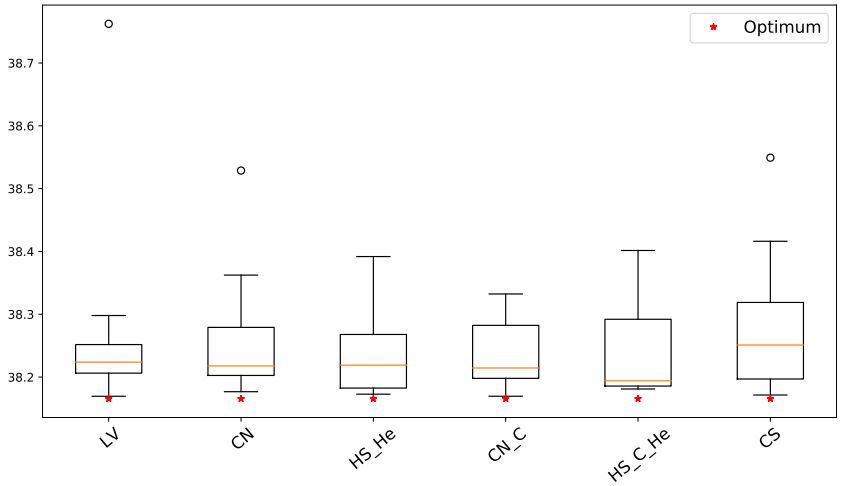


FIGURE 4.10: Comparison of the convergence value of various discrete kernels during the BO of the mixed-variable Goldstein function over 20 repetitions. Focus on the mixed-variable kernels.

where g_0 is the gravitational acceleration at the Earth's surface (set to 9.80665 m/s^2), I_{sp} is the specific impulse of the engine, M_i and M_f are respectively the initial and final mass of the rocket. Different organizations of the solid propellant in the cylindrical motor case may be defined depending on the grain geometry inside the case, which has consequences on the level of thrust of the solid rocket engine along the trajectory. A star grain configuration is used in the simulation as it presents the advantage of providing a constant propellant burning surface along the trajectory and by consequence a constant thrust. Three different engine options, each one with different efficiency and geometrical/physical properties, are considered. In the propulsion discipline, the propellant burning rate is computed as a function of the type of propellant (four options are considered: Butalite, Butalane, Nitramite and pAIM-120), the propellant mass M_{prop}, combustion chamber pressure P_{comb} and nozzle throat diameter D_t:

$$\dot{m} = \frac{M_{prop}}{\Delta t} = \frac{P_{comb}\pi D_t^2}{4c^*} \tag{4.30}$$

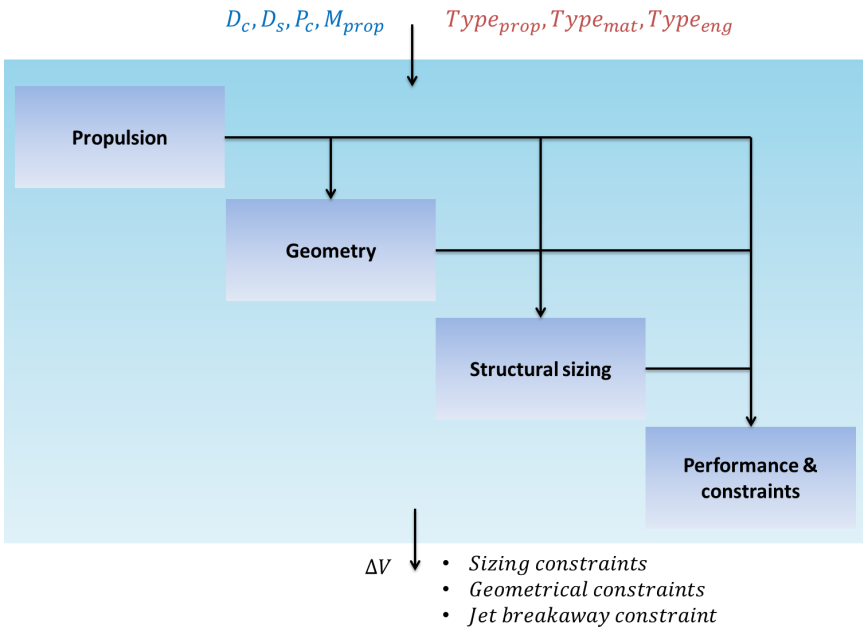


FIGURE 4.11: Multidisciplinary design analysis for a solid rocket engine.

Variable	Nature	Min	Max	Levels
D_t - Nozzle throat diameter [m]	continuous	0.2	1	[-]
D_e - Nozzle exit diameter [m]	continuous	0.5	1.2	[-]
P_{comb} – Chamber pressure [bar]	continuous	5	300	[-]
M_{prop} – Propellant mass [kg]	continuous	2000	15000	[-]
$Type_{prop}$ - Type of propellant	discrete	[-]	[-]	Butalite, Butalane, Nitramite, pAIM-120
$Type_{mat}$ - Type of material	discrete	[-]	[-]	Aluminium, Steel
$Type_{eng}$ - Type of engine	discrete	[-]	[-]	type 1, type 2, type 3

TABLE 4.2: Variables characterizing the solid rocket engine test-case

where T is the engine thrust, Δt is the thrust duration and c^* is the characteristic velocity associated to the selected type of propellant. The burning rate can then be used in order to compute the specific impulse, as follows:

$$I_{sp} = eff\frac{T}{\dot{m}} = \varepsilon\frac{T\Delta t}{M_{prop}} \tag{4.31}$$

where ε is an efficiency factor related to the selected type of engine (another design variable). To avoid jet breakaway (schematically represented in Figure 4.12), a constraint on the gas expansion is considered. In the mass budget and geometry design discipline, the masses of the solid rocket

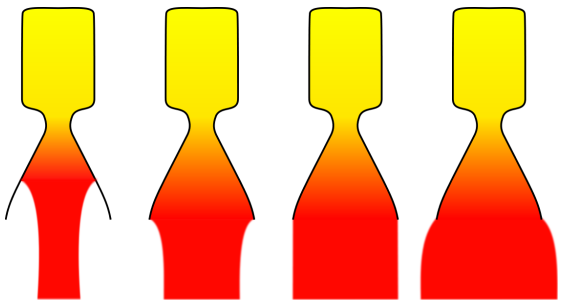


FIGURE 4.12: Under or over gas expansion at the nozzle exit

engine case and the nozzle are estimated based on the type of material, the nozzle geometry and the propellant masses. The total initial mass is computed as the sum between the propellant mass M_{prop}, the casing mass M_c and the mass of the auxiliary equipment M_{aux}:

$$M_i = M_{prop} + M_c + M_{aux} \tag{4.32}$$

The casing mass depends on the selected type of engine as well as on the combustion chamber pressure, and it is computed as:

$$M_c = \rho t_c D_c^2 \pi (1 + L_c/D_c) + \rho t_c \pi D_c^2 \tag{4.33}$$

where L_c, D_c and t_c are respectively the length, diameter and thickness of the cylindrical-shaped casing, while ρ represents its density, which depends on the selected type of engine. The thickness of the case must be determined so that it can safely withstand the combustion pressure:

$$t_c = \frac{P_{comb} D_c}{2\sigma_u} \tag{4.34}$$

where σ_u is the ultimate tensile strength of the selected material. Finally, the auxiliary mass is assumed to be proportional to the sum of the propellant and casing mass. More specifically it is computed as:

$$M_{aux} = (M_{prop} + M_c)\frac{5}{100} \tag{4.35}$$

Six constraints relative to the available volume for the propellant, the case geometry and the nozzle geometry are present in the problem. Finally, the structure discipline includes a constraint representing the maximal material stress, as the motor case must be able to withstand the maximum combustion chamber pressure under any possible operating condition. The performance and constraints module estimates the speed increment which combines the rocket engine thrust, the specific impulse, the propellant mass and the booster dry mass. The sounding rocket optimization test-case described above can be formally defined as:

$$\begin{aligned} \min \quad & -\Delta V(D_t, D_e, P_{comb}, M_{prop}, Type_{prop}, Type_{mat}, Type_{eng}) \\ \text{w.r.t.} \quad & D_t, D_e, P_{comb}, M_{prop}, Type_{prop}, Type_{mat}, Type_{eng} \\ \text{s.t.:} \quad & g_i(D_t, D_e, P_{comb}, M_{prop}, Type_{prop}, Type_{mat}, Type_{eng}) \leq 0 \quad \text{for i } = 1, ..., 8 \end{aligned} \tag{4.36}$$

with:

$$\begin{gathered} D_t \in [0.2, 1], \quad D_e \in [0.5, 1.2], \quad P_{comb} \in [5, 300], \quad M_{prop} \in [2000, 15000], \\ Type_{prop} \in \{0, 1, 2, 3\}, \quad Type_{mat} \in \{0, 1\}, \quad Type_{prop} \in \{0, 1, 2\} \end{gathered}$$

For the SMBDO of this thrust performance optimization problem, an initial training data set of 72 samples is used (*i.e.,* 3 samples for each independent GP in the CW case) and subsequently 150 additional data points are infilled during the optimization process. The results obtained for the optimization of the sounding rocket engine ΔV over 10 repetitions are shown in Figures 4.13 and 4.14.

These results show a fairly similar convergence rate for all of the compared kernel up to 90 infilled data samples, at which point the CS and LV kernels are able to better refine the solution in the optimum neighborhood. Moreover, at convergence, the LV kernel provides a better and more consistent optimum value when compared to the other 3 kernels.

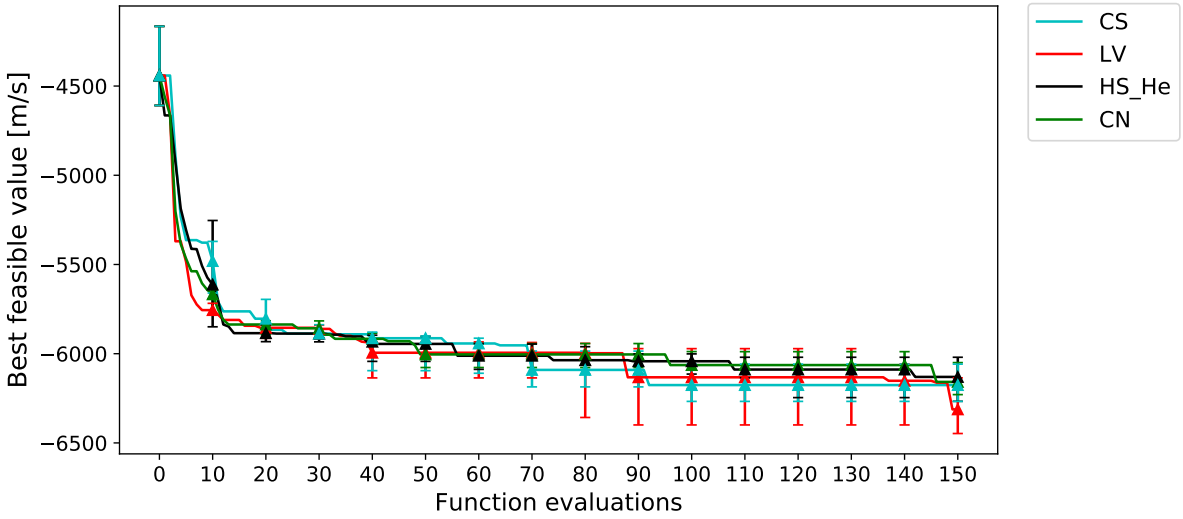


FIGURE 4.13: Comparison of the convergence rate of various discrete kernels during the BO of the launch vehicle propulsion performance optimization over 10 repetitions.

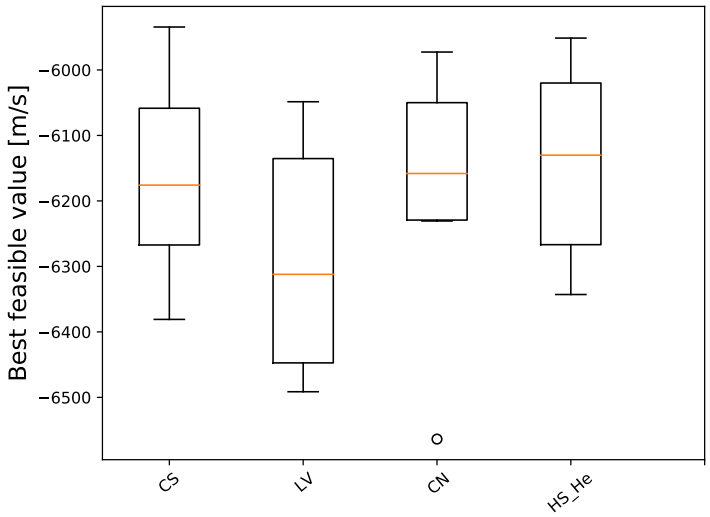


FIGURE 4.14: Comparison of the convergence value of various discrete kernels during the BO of the launch vehicle propulsion performance optimization over 10 repetitions.

4.3.6 Launch vehicle thrust frame design optimization

The second aerospace design problem considered in this chapter is the optimization of the launch vehicle thrust frame, briefly presented in Chapter 3. More specifically, a former version of the Ariane 6 aft bay is considered. As previously mentioned, the thrust frame is located at the bottom of the launcher first stage and has the purpose of withstanding the weight of the launcher as well as the thrust of the two solid rocket boosters during the lift-off phase. This structure is composed of a cylindrical outer skin, stiffened by frames and stringers, and an inner body comprising three major frames and an inner skin. The static loads that are considered for this design problem are the longitudinal and lateral thrust of the two solid rocket boosters. The boundary conditions are defined by modeling the first stage composite bottom skirt clamped to the upper interface of the bottom skirt.

The objective of the considered design problem is to minimize the overall mass of the thrust frame, thus improving the overall launch vehicle performance, while complying with structural

integrity constraints. The considered design variables are the skins thicknesses t_i of 12 regions in which the bottom skirt is decomposed as well as of the number of stringers N_s and the number of frames N_f. For illustrative purposes, two possible bottom skirt configurations are provided in Figure 4.15. The 12 thicknesses are continuous variables while the number of stringers and

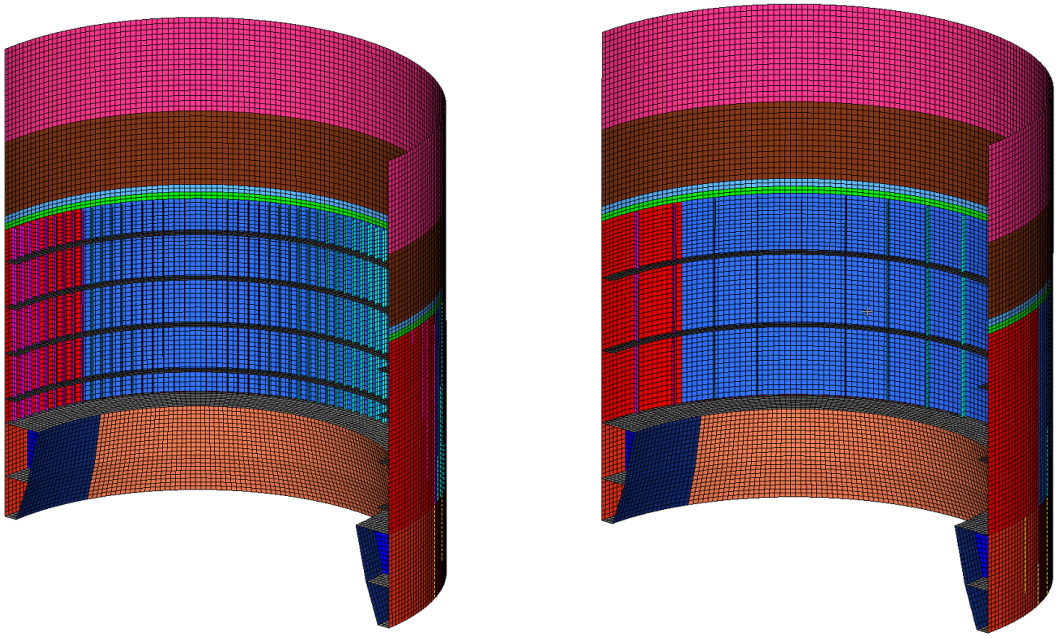


FIGURE 4.15: Two examples of possible bottom skirt configurations. On the left figure 144 stringers and 4 frames are included while on the right one only 36 stringers and 2 frames are present.

frames are discrete variables, each one characterized by 3 possible values, thus resulting in a total of 9 combinations (*i.e.,* categories). For the sake of simplicity, the same range is considered for all the thicknesses variables characterizing the problem. More specifically, the minimum and maximum bounds are 1 and 30 mm, respectively, due to manufacturing constraints. A summary of the variables characterizing the design problem is provided in Table 4.3. The considered design

Variable	Nature	Min	Max	Possible values
$t_{1,12}$ [mm]	continuous	1	30	[-]
N_s	discrete	[-]	[-]	36, 72, 144
N_f	discrete	[-]	[-]	2, 4, 8

TABLE 4.3: Variables characterizing the bottom skirt structural analysis test-case

problem is subject to several constraints necessary to ensure the structural integrity of the bottom skirt during the lift-off phase. The Von Mises stresses [40] computed on the triangular (VM^t) and quadrilateral (VM^q) mesh elements of both the inner and outer skins of the structure must be inferior to a given limit value (VM^{max}, which is equal to the ultimate tensile strength of the material plus a safety margin). In a similar way, the maximal upper interface longitudinal overflux (F), given by the maximal force at the interface, must also be constrained to a given limit value (F_{max}). This results in a total of 5 constraints that must be satisfied for a candidate

solution to be feasible. This design problem can be formulated as follows:

$$\begin{aligned}
\textbf{min} \quad & \text{Mass}(t_1, \dots, t_{12}, N_s, N_f) \\
\text{w.r.t.} \quad & t_1, \dots, t_{12} \in [1, 30] \\
& N_s \in \{36, 72, 144\} \\
& N_f \in \{2, 4, 8\} \\
\text{s.t.:} \quad & VM^t_{inner}(t_1, \dots, t_{12}, N_s, N_f) - VM^{max}_{inner} \leq 0 \\
& VM^q_{inner}(t_1, \dots, t_{12}, N_s, N_f) - VM^{max}_{inner} \leq 0 \\
& VM^t_{outer}(t_1, \dots, t_{12}, N_s, N_f) - VM^{max}_{outer} \leq 0 \\
& VM^q_{outer}(t_1, \dots, t_{12}, N_s, N_f) - VM^{max}_{outer} \leq 0 \\
& F(t_1, \dots, t_{12}, N_s, N_f) - F_{max} \leq 0
\end{aligned} \tag{4.37}$$

In order to evaluate the objective and constraint functions values of the considered design problem, the MSC Nastran FEM software [82] is used. In practice, a separate finite element model is created for every considered combination of number of stringers and frames due to the need of a distinct meshing for every configuration. A number of static FEM analyses is then performed with varying skins thicknesses, according to the values present in the data sets, and using the finite element model corresponding to the discrete case the considered sample belongs to. The lift-off conditions are simulated by considering two aligned vertical loads F_x and two opposing horizontal loads F_y, applied on the side of the bottom skirt, as illustrated in Figure 3.20. The compared BO techniques are initialized with a data set of 45 samples (*i.e.,* 5 samples for each independent GP in the CW case) and subsequently 150 additional data points are infilled during the optimization process. The results obtained for the optimization of the thrust frame design over 10 repetitions are shown in Figures 4.16 and 4.17. Please note that the feasible mass values are normalized due to the proprietary nature of the finite element model. Also in this case, the compared kernels behave similarly in the first half of the optimization process. In the second part of the optimization, instead, it can be seen that the two simpler and linear kernels, *i.e.,* LV and CS, yield a faster and more consistent convergence when compared to more complex kernels, such as the hetersocedastic HS and CN ones. This can first of all be explained by the linear relations existing between the responses associated to the various levels of the 2 discrete variables. Indeed, the different responses (in terms of both mass and structural stresses) which are obtained by varying the number of structural reinforcements (*i.e.,* stringers and frames) are expected to be linearly dependent. As a consequence, kernel parameterizations which define the covariance between levels in a linear fashion are expected to provide a better modeling of the problem functions in case a small training data set is provided. Nevertheless, even in this context, the LV kernel is expected to provide slightly better results than the CS one when dealing with discrete variables characterized by more than 2 levels, which is not coherent with the obtained results. This result is most likely due to the convergence of the GP training processes towards local optima caused by a non-ideal initialization of the hyperparameters. As mentioned in Chapter 3, this issue can technically be avoided by providing multiple random initializations to the gradient based marginal likelihood optimization. However, this solution tends to considerably increase the computational overhead of the BO process, thus rendering the time required to produce the necessary results much longer.

Finally, for illustrative purposes the best optimum (*i.e.,* minimum mass complying with structural integrity constraints) obtained among all the performed repetitions is presented in Figure 4.18 in terms of the thicknesses characterizing the solution. Additionally, the spatial distribution and value of the structural integrity constraints is provided in Figure 4.19. The left part of Figure 4.19 shows that the maximum Von Mises constraint values are obtained in the junction area between the bottom skirt and the lateral boosters. These mechanical stress peaks are located

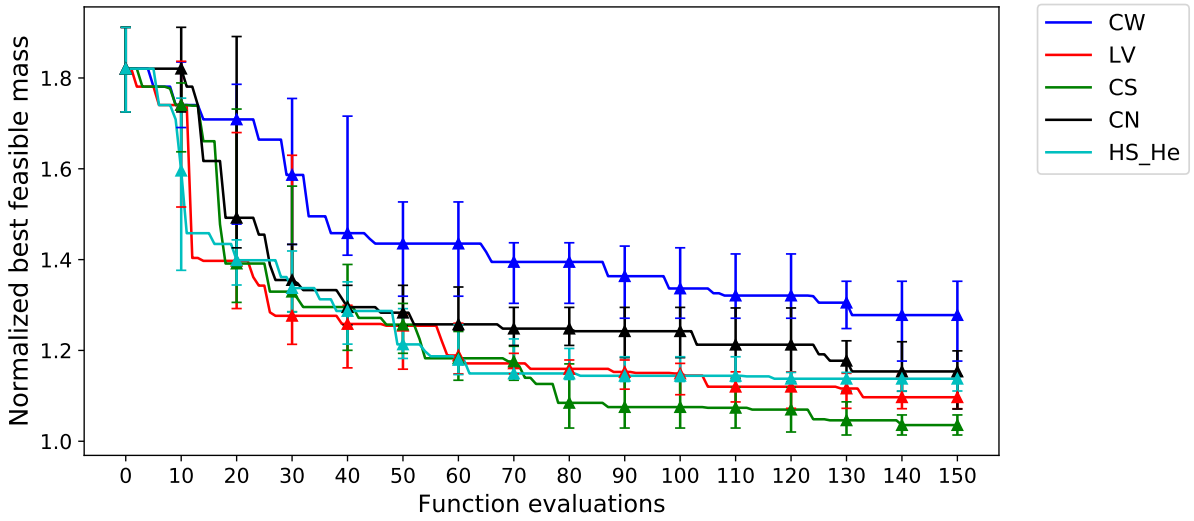


FIGURE 4.16: Comparison of the convergence rate of various discrete kernels during the BO of a launch vehicle thrust frame design over 10 repetitions.

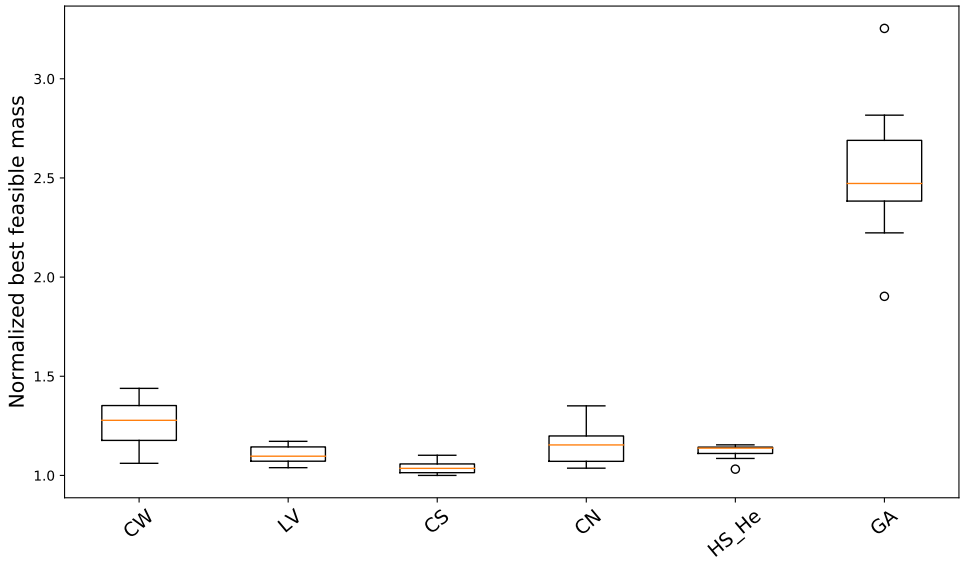


FIGURE 4.17: Comparison of the convergence value of various discrete kernels during the BO of launch vehicle thrust frame design over 10 repetitions.

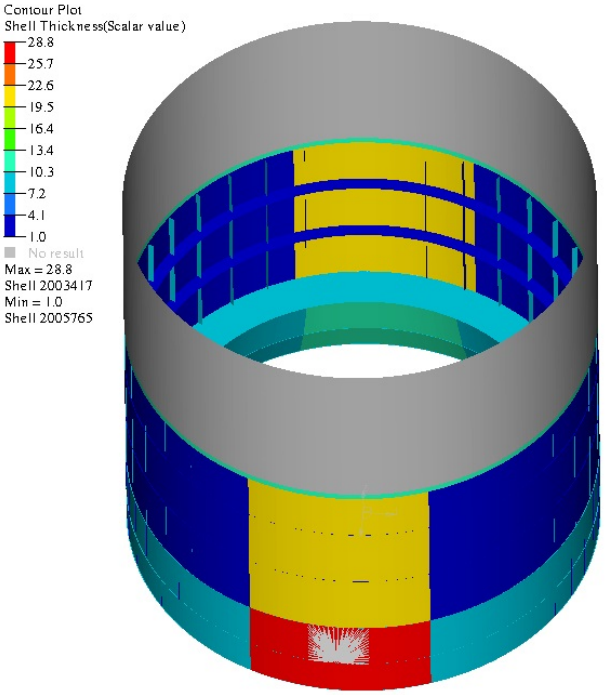


FIGURE 4.18: Skin thicknesses characterizing the optimal solution obtained for the design optimization of the launch vehicle thrust frame.

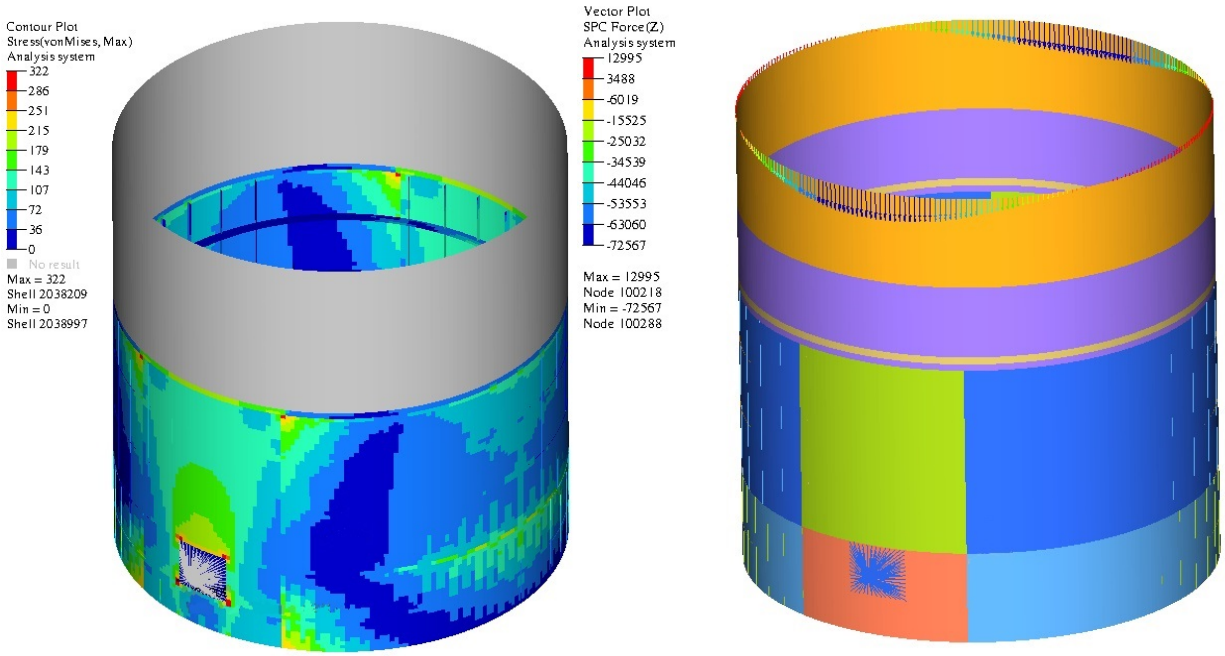


FIGURE 4.19: Von Mises stress (left) and overflux (right) characterizing the optimal solution obtained for the design optimization of the launch vehicle thrust frame.

close to the rigid static load application surface and are caused by the bending induced by the booster thrust on the aft bay. As a consequence, the optimal skin thicknesses are larger close to the previously mentioned load application surface (*i.e.,* between 20 and 30 mm) in order to ensure structural integrity. The central section of the bottom skirt, instead, presents the lower constraint values, together with the reinforcement frames and stringers. As a consequence, the

optimal skin thicknesses associated to these areas of the structure are small (*i.e.,* between 1 and 2.5 mm). Similarly to the Von Mises Stress, the largest obtained overflux constraint value is located vertically above the junction area between the bottom skirt and the lateral boosters, as can be seen from the right part of Figure 4.19. However, the overflux constraint is not saturated as its maximum value does not reach the structural constraint threshold. Overall, it can be stated that the driving parameter of this optimization is the Von Mises stress, as neither the overflux constraint nor the thickness design variables converge to one their respective bounds. Furthermore, the presented optimum is located in the problem category characterized by the smallest number of reinforcements (*i.e.,* 36 stringers and 2 frames), and it could therefore be worth including a non-reinforced category to the problem for further analyses. Finally, in order to improve the overall structure mass, it could also be interesting to refine the initial division of the bottom skirt in constant thickness sections in order to determine a better parameterization of the optimization problem.

4.3.7 Result synthesis

Overall, the results obtained on both analytical and engineering related benchmarks show that the proposed mixed-variable BO algorithm allows provide a considerably faster convergence speed with respect to both the standard engineering approach (*i.e.,* a separate surrogate model per category) and mixed-variable heuristic optimization methods, such as the GA. Indeed, for a part of the considered test-cases (*e.g.,* Branin and Goldstein), the proposed methods consistently converge to the neighborhood of the considered problem optimum, regardless of the considered kernel parameterization, whereas the reference methods would require a large number of additional function evaluations in order to provide a similar performance. It is also interesting to note that the difference in performance between the considered discrete kernel parameterizations in terms of convergence speed as a function of the considered problem characteristics tends to be less important when compared to the differences in modeling performance which are shown in Chapter 3. This is particularly noticeable for simpler and lower dimension test-cases, as can for instance be seen for the Branin and Goldstein benchmarks. Furthermore, the obtained results also show that the best performing kernels for a given function in terms of modeling accuracy do not necessarily also yield the best performance in terms of convergence speed, as can for instance be noticed for the augmented Branin function, where both the level-wise and category-wise hypersphere decomposition kernels perform slightly worse than the respective coregionalization kernels, which is the opposite of what is obtained from the modeling performance analysis.

The difference between modeling and optimization performance can first of all be explained by the fact that the modeling performance with respect to the constraints the considered problem is subject should also be taken into account in order to accurately assesses the dependence on the kernel parameterization. However, the functions which constrain the analytical test-cases are fairly simple and smooth, and should therefore be easily modeled by the considered mixed-variable GP, regardless of the considered kernel parameterization.

A second explanation for the difference between modeling and optimization performance is that along the BO process, depending on the location of the infilled data samples, the marginal likelihood landscape might become multi-modal. By consequence, the GP training for the more complex discrete kernels, such as the coregionalization one, might require a large number of random initialization in order to ensure its converge. On the contrary, the training of simpler kernels, such as the CS one is more robust with respect to the training initialization, which for instance explains its good performance in the launch vehicle thrust frame design test-case. Furthermore, during the exploitation phases the BO algorithms tend to infill a large number of samples within a few neighborhoods which are considered to be most promising. As a consequence, the GP training tends to be driven by the local trends in these neighborhoods rather than by the modeled function global trend. The same phenomenon can therefore also occur in the mixed-variable

case, where a large amount of data samples infilled within a few categories of the considered problem might worsen the overall modeling performance of the GP along the BO process. Also in this case, simpler models should be more robust with respect to this phenomenon thanks to the impossibility of being driven by the need to model local trends. For instance, the training of a heteroscedastic GP might be driven by a large number of data samples infilled within a heteroscedastic area of an overall homoscedastic level or category, whereas the same cannot happen when considering a simpler homoscedastic model (*i.e.,* LV kernel). Finally, it is worth noticing that although the compared BO algorithms are provided with small sized data samples at the beginning of the optimization process, their relative performance seems to be more dependent on the particular characteristics of the considered kernel (*e.g.,* negative covariance values and heteroscedasticity) than on the number of hyperparameters necessary to represent the kernel, if compared to the results obtained in Chapter 3.

4.4 Conclusions

In this chapter, an extension of Bayesian Optimization methods based on the use of discrete kernels and allowing to solve mixed-variable problems is proposed. Commonly used infill criteria, namely the Expected Improvement, Probability of Feasibility and Expected Violation are shown to be valid in a mixed-variable context as long as the GP kernels are properly parameterized. Finally, the optimization of the resulting acquisition functions in the mixed-variable design space is discussed. The mixed-variable BO methods are then applied to a number of analytical and aerospace design related test-cases. Overall, the obtained results show that the proposed algorithms provide a considerably faster convergence to the considered problems optima with respect to the reference methods. This allows to show that by modeling mixed-variable problem functions with the help of a single mixed-variable kernel rather than multiple purely continuous ones, it is possible to better exploit the information provided by the training data set and by consequence reduce the number of functions evaluations necessary to converge towards the optimum of a given problem. Moreover, the results also allow to notice a difference in relative performance among the considered discrete kernel parameterizations depending on whether a modeling or optimization framework is considered. This difference is mainly due to the specific behavior of the various kernels when dealing with training data sets containing a large part of the samples located within a few neighborhoods, corresponding to the most promising areas of the design space.

The proposed mixed-variable BO technique provides a solution for the optimization of constrained problems characterized by functions depending simultaneously on continuous and discrete variables. However, this approach does not allow to deal with Variable-size Design Space Problems in their most generic formulation, as defined in Chapter 2, in which the design space and the feasibility domain vary dynamically along the optimization problem as a function of the dimensional variable values. In Chapter 5, 2 extensions of the mixed-variable BO discussed in this chapter allowing to solve VSDSP are proposed, discussed and tested on different test-cases.

CHAPTER 5

Bayesian optimization of variable-size design space problems

5.1 Introduction

In Chapter 3 and Chapter 4, the modeling and optimization of problems depending simultaneously on continuous and discrete variables by relying on Gaussian processes are presented and discussed. However, as is detailed in Chapter 2, the design problems of actual complex systems often present the necessity of choosing between different possible technological and/or architectural alternatives, which can be described with the help of different specific design variables and subject to different constraints. This characteristic translates into problems characterized by a design space and a feasibility domain which vary dynamically along the optimization process as a function of so-called dimensional variables, which represent such design choices. This kind of optimization problems are sometimes referred to as Variable-size Design Space Problems (VSDSP) [93]. An example of such an optimization problem is the preliminary design of launch vehicles, for which the performance criteria and constraints depend on continuous variables (*e.g.,* sizing parameters) as well as on discrete design parameters (*e.g.,* type of propulsion, choice of materials). Furthermore, depending on dimensional variables such as the type of propulsion and propellants, different kinds of continuous and discrete variables may need to be optimized, thus resulting in a dynamically varying design space and feasibility domain. For illustrative purposes, a few of the examples of dimensional variables and associated continuous and discrete design variables and constraints which may be encountered within the context of RLV design presented in Chapter 2 are provided anew below:

- Propulsion

  Different types of propulsion can be considered for each stage of a launch vehicle: solid, liquid and hybrid. Due to the considerably difference in nature between these technologies, the design parameters associated to the propulsion sub-system can be very different. Notable examples are the type of reductant and oxydant (*i.e.,* propellant) the engine combustion relies on. Depending on the selected type of propulsion, the set of possible propellant choice differs. Additionally, each type of propulsion is associated to a set of technology specific variables. For instance, if solid propulsion is considered, the shape of the propellant grain (*i.e.,* circular, star-shaped) as well as the specific shape sizing must be optimized.

- Material

  The structure of the various sub-systems of a launch vehicle can be either made of metallic or composite material. While the first choice is mainly associated to a specific material

and the related geometrical sizing parameters, composite materials can be associated to the matrix and fiber choices as well as continuous parameters such as the orientation of each composite material ply. Furthermore, the number and type of structural integrity constraints associated each choice of material are also considerably different.

- Flight configuration

  Among the alternative solutions which can allow to re-use launch vehicles, one of the options consists in including lifting surfaces in the system design, thus enabling the launch vehicle to glide back to the landing site. If the option of including a lifting surface in the design is selected, a number of design variables associated to these lifting surface, such as shape and sizing parameters, must be optimized. Furthermore, additional constraints necessary to ensure the structural integrity of the lifting surfaces must be complied with.

A generic VSDSP can be defined as follows:

$$
\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}) \qquad f : \mathbb{R}^{n_x(\mathbf{w})} \times \prod_{d=1}^{n_z(\mathbf{w})} F_{z_d} \times F_w \rightarrow F_f \subseteq \mathbb{R} \\
\text{w.r.t.} \quad & \mathbf{x} \in F_x(\mathbf{w}) \subseteq \mathbb{R}^{n_x(\mathbf{w})} \\
& \mathbf{z} \in \prod_{d=1}^{n_z(\mathbf{w})} F_{z_d} \\
& \mathbf{w} \in F_w \\
\text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0 \\
& g_i : F_{x_i}(\mathbf{w}) \times \prod_{d=1}^{n_{z_i}(\mathbf{w})} F_{z_{d_i}} \times F_w \rightarrow F_{g_i} \subseteq \mathbb{R} \qquad \text{for} \qquad i = 1,...,n_g(\mathbf{w})
\end{aligned}
\tag{5.1}
$$

As discussed in Chapter 2, the continuous and discrete search spaces characterizing the problem functions described in Eq. 2.1 are not fixed, as they depend on the values of the dimensional variables. However, although the design space may vary, the physical quantity that is represented by the problem objective function remains the same throughout the entire optimization process. In the same way, the feasibility domain also varies according to the values of $\mathbf{w}$, which results in different candidate solutions being subject to different constraints (in terms of type and number of active constraints). Furthermore, it is important to highlight that the constraint functions as well present a search space which can vary as a function of the dimensional variables values. By consequence, the optimization problem defined in the equations above varies dynamically along the optimization problem, both in terms of number and type of design variables as well as feasibility, depending on the values of the candidate solution dimensional variables.

For the sake of simplicity, the possibility of having the dimensional variable search space vary as a function of the dimensional variables themselves is not taken into consideration. However, it can be easily shown that these particular cases can always be reformulated under the formalism of Eq. 2.1 by considering the resulting combinatorial design variable search space.

Although VSDSP represent the majority of actual complex system design problems, very few optimization methods or algorithms allowing to solve such problems in their generic formulation exist. In most cases, the chosen approach in order to solve VSDSP consists in decomposing the global problem into several fixed-size sub-problems (*i.e.,* one per combination of dimensional variable values) and separately optimizing each one of them. Although straightforward and easily implementable, this approach tends to quickly become computationally overly demanding when confronted with VSDSP characterized by a large number of sub-problems, especially in the presence of computationally intensive objective and/or constraint functions. Alternatively,

a few algorithms have been extended in order to deal with VSDSP. In [3] a so-called hidden gene adaptation of GA is proposed for the optimization of inter-planetary trajectories with variable number of gravitational assists, resulting in the appearance and disappearance of variables describing these orbital maneuvers. In the proposed algorithm, each candidate solution is represented by a chromosome containing the entirety of genes that can characterize the problem at hand. However, not all the genes are taken into account when computing the value of the objective and constraint functions. The choice regarding which genes are considered and which genes are 'hidden' depends on the values of a limited number of so-called activation genes. This variant of GA has the advantage of being intuitive and easily implementable. However, cross-overs and mutations over hidden (*i.e.,* with no influence on the problem functions) parts of the vector result in ineffective numerical operations, thus wasting computational effort. Furthermore, for problems characterized by a large number of discrete categories, the vector containing the entirety of possibly present variables might become considerably large and therefore inefficient memory-wise. In the same way, the authors also implemented the hidden gene approach within DE for a similar inter-planetary trajectory planning problem in [2]. A more complex, but theoretically more efficient adaptation of GA called Structured-Chromosome Evolutionary Algorithm is proposed by H.M. Nyew *et al.* in [93]. In this algorithm, the candidate solution is conceptually represented by a hierarchical multi-level chromosome structure rather than a linear one. Differently than in the standard formulation of GA, the genes of each chromosome are linked by both vicinity and hierarchy relationships. For this reason, the encoding of the single variables becomes more complex as it also includes pointers to both the immediate neighbor at the same level and pointers to eventual children genes. Furthermore, in case the mutation of a dimensional variable gene results in the appearance of additional design variables, it is necessary to ensure that the newly created genes are of the correct type and comply with the problem specifics. Compared to the hidden genes approach, this solution has the advantage of not performing computationally wasteful mutations and cross-overs. However, the encoding of the individual chromosomes is much more complex and often requires problem-specific knowledge in order to be efficiently implemented.

A second family of algorithms which have been extended in order to provide a solution to VSDSP are mesh-based optimization algorithms, as is discussed in [4], [5], [9] and [79]. Although each paper describes a different algorithm, they share the same approach to the optimization problem which consists in an alternation between an optional user-defined search phase and a poll phase, in which an exhaustive and schematic optimum search over a mesh is performed. When applying this family of algorithms to VSDSP, it is necessary to take into account the appearance and disappearance of design variables as a function of the dimensional variables characterizing the incumbent solution around which the mesh is centered. In order to solve this issue, the mesh algorithms mentioned above alternate between searches over a mesh defined in the dimensional design space and searches over a mesh defined in the continuous and discrete search space dependent on the dimensional variables values. The application of this algorithm requires the user to provide the definition of the neighborhood of a candidate solution in the dimensional design space in order to create the mesh. The alternative approach consists in defining the mesh as the entirety of the dimensional search space, which quickly becomes unfeasible when dealing with computationally intensive problems with large combinatorial design spaces.

Both population and mesh-based algorithms for the solution of VSDSP tend to require a large number of function evaluations in order to converge to the considered problem optimum. Furthermore, they also rely on a penalization-based handling of constraints, which can often be inefficient if the penalization weights are not properly tuned, especially in the presence of a large number of constraints. For these reasons, in this chapter two alternative BO approaches allowing to solve VSDSP with a lower number of function evaluations with respect to the existing alternatives are proposed. The first one is based on the separate and independent optimization of every sub-problem characterized by a fixed-size design space coupled with a budget allocation strategy relying on the information provided by the surrogate models of the various sub-problems

functions. The second method, instead, is based on the definition of a Gaussian Process kernel defined in the variable-size design space, allowing to compute the covariance between data samples which belong to a partially different design space. Following this introduction, in the second Section the budget allocation strategy is presented, and the criteria allowing to allocate the computational budget between sub-problems and discard non-optimal sub-problems are discussed. In Section 3, two alternative definitions of a GP kernel allowing to compute the covariance between data samples belonging to partially different design spaces are proposed. Subsequently, in Section 4 the two aforementioned methods for the optimization of VSDSP are applied on both analytical and engineering related test-cases and the obtained results are presented and discussed. Finally, in Section 5 the relevant conclusions which can be drawn from the obtained results are provided and possible perspectives and improvements related to the proposed algorithms are discussed.

5.2 Budget allocation strategy

As mentioned in the introduction, due to the limitations of the existing algorithms allowing to solve VSDSP, the most commonly used approach consists in decomposing the global problem into N_p fixed-size mixed continuous/discrete sub-problems which can be independently optimized by relying on more standard algorithms, such as the mixed-variable BO presented in Chapter 4. In practice, each sub-problem q can be obtained by fixing the dimensional variables $\mathbf{w}$ to a given set of possible values $\mathbf{w}_q$ and is formulated as follows:

$$
\begin{aligned}
\min \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}_q) \qquad f : \mathbb{R}^{n_x(\mathbf{w}_q)} \times \prod_{d=1}^{n_z(\mathbf{w}_q)} F_{z_d} \times F_w \rightarrow F_f \subseteq \mathbb{R} \\
\text{w.r.t.} \quad & \mathbf{x} \in F_x(\mathbf{w}_q) \subseteq \mathbb{R}^{n_x(\mathbf{w}_q)} \\
& \mathbf{z} \in \prod_{d=1}^{n_z(\mathbf{w}_q)} F_{z_d} \\
\text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{z},\mathbf{w}_q) \leq 0 \\
& g_i : F_{x_i}(\mathbf{w}_q) \times \prod_{d=1}^{n_{z_i}(\mathbf{w}_q)} F_{z_{d_i}} \times F_w \rightarrow F_{g_i} \subseteq \mathbb{R} \qquad \text{for} \qquad i = 1,...,n_g(\mathbf{w}_q)
\end{aligned}
\tag{5.2}
$$

The optimization problem presented in Eq. 5.2, obtained by fixing the values of $\mathbf{w}$ to $\mathbf{w}_q$ (*i.e.,* one of $\mathbf{w}$ categories), depends on a fixed number of continuous and discrete variables and is subject to a fixed number of constraints, and can therefore be solved by relying on the mixed-variable BO described in Chapter 4. Alternatively, the sub-problems can be enumerated by regrouping all the dimensional variables as a single variable w defined in the combinatorial space of $\mathbf{w}$. In this case, each sub-problem is simply defined by one of the possible levels of the scalar variable w. It is important to remember that although the various sub-problems defined as in Eq. 5.2 can be characterized by a different set of design variables as well as different constraints, their objective function always models the same quantity with the same unit of measure (*e.g.,* gross lift-off weight or total cost of a launch vehicle), as otherwise the comparison between sub-problems would lose meaning.

In order to determine the global optimum of a VSDSP as defined in Eq. 2.1, a separate and independent sub-problem optimization must be performed for every possible category of $\mathbf{w}$. As a consequence, this approach quickly becomes computationally inefficient, if not unfeasible, when dealing with optimization problems which present a large dimensional variable combinatorial search space, due to the fact that all the non-optimal sub-problems must be optimized until convergence regardless of the provisional results obtained along the optimization process. This issue becomes particularly problematic when dealing with computationally intensive problems,

which are particularly common within the context of complex system design. In order to partially avoid this issue when dealing with VSDSP, the first approach for the optimization of variable-size design space problems which is proposed in this chapter is based on coupling the separate and independent optimization of each sub-problem defined by Eq. 5.2 with a computational budget allocation strategy. For the sake of conciseness, this approach is referred to as Strategy for the Optimization of Mixed Variable-Size design space Problems (SOMVSP). The underlying idea is to rely on the information provided by the surrogate models of each sub-problem objective and constraint functions along the optimization process in order to determine which sub-problems are more likely to contain the global optimum. At every SOMVSP iteration, this information provided by the various sub-problems surrogate models is exploited in order to allocate a different computational budget to each sub-problem (in terms of infilled data samples) proportionally to how promising it is. Moreover, in order to further optimize the usage of the overall computational budget, the sub-problems least likely to contain the global optimum can also discarded between SOMSVP iterations. In synthesis, the proposed strategy relies on two main axes: the allocation of a different optimization computational budget to each sub-problem as a function of the predicted optimal solution (in terms of both objective function value and feasibility), and the possibility of discarding a given sub-problem in case its predicted relative performance with respect to the other sub-problems crosses a given threshold. These two mechanisms are detailed in the following paragraphs.

5.2.1 Discarding of non-optimal sub-problems

The proposed computational effort allocation logic relies on both the dimension of the considered sub-problem and its predicted relative performance with respect to the objective function value as well as the solution feasibility. At each SOMVSP iteration, the relative performance of each sub-problem q with respect to the others must be determined. In order to do so, three different predicted optima are computed for each sub-problem by considering different confidence interval scenarios: the *Best Case* (BC), the *Worst Case* (WC) and the *Nominal Case* (NC). In practice, these scenarios correspond to the **predicted** feasible optimum value of the sub-problem q by considering an optimistic, pessimistic and null value of the predicted variance, respectively, and they are defined as follows:

$$\begin{aligned} BC_q &= \min \hat{y}_q(\mathbf{x}, \mathbf{z}, \mathbf{w}_q) - a \cdot \hat{s}_q(\mathbf{x}, \mathbf{z}, \mathbf{w}_q) \\ &\text{s.t. } EV\left(g_c(\mathbf{x}, \mathbf{z}, \mathbf{w}_q)\right) < t_c \ for \ c = 1, ..., n_{g_i}(\mathbf{w}_q) \end{aligned} \tag{5.3}$$

$$\begin{aligned} WC_q &= \min \hat{y}_i(\mathbf{x}, \mathbf{z}, \mathbf{w}_q) + a \cdot \hat{s}_q(\mathbf{x}, \mathbf{z}, \mathbf{w}_q) \\ &\text{s.t. } EV\left(g_c(\mathbf{x}, \mathbf{z}, \mathbf{w}_q)\right) < t_c \ for \ c = 1, ..., n_{g_i}(\mathbf{w}_q) \end{aligned} \tag{5.4}$$

$$\begin{aligned} NC_q &= \min \hat{y}_q(\mathbf{x}, \mathbf{z}, \mathbf{w}_q) \\ &\text{s.t. } EV\left(g_c(\mathbf{x}, \mathbf{z}, \mathbf{w}_q)\right) < t_c \ for \ c = 1, ..., n_{g_i}(\mathbf{w}_q) \end{aligned} \tag{5.5}$$

where $a \in \mathbb{R}^+$ is a tunable parameter representing how conservative are the definitions of the BC and WC scenarios. In practice, a represents the confidence the user gives to the GP models modeling accuracy of the various sub-problems functions. In general, larger values of a tend to result in more conservative and robust results, however, they also tend to reduce the convergence speed of the proposed SOMVSP. For the sake of simplicity, only values of $a = 2$ and $a = 3$ are considered in this chapter, which approximately correspond to confidence intervals of 95% and

99%. It can be noted that the constraints are handled by relying on the EV rather than on the PoF. This choice is due to the fact that the the PoF would tend to penalize the sub-problems characterized by larger number of constraints.

For illustrative purposes, the following unconstrained one-dimensional continuous function is considered:

$$f(x) = \cos(x)^2 + \sin(x) + 0.25 \tag{5.6}$$

A GP model of this function is created by relying on 7 data samples, and the Best-case, Worst-case and Nominal-case functions associated to the given model are provided in Figure 5.1. In

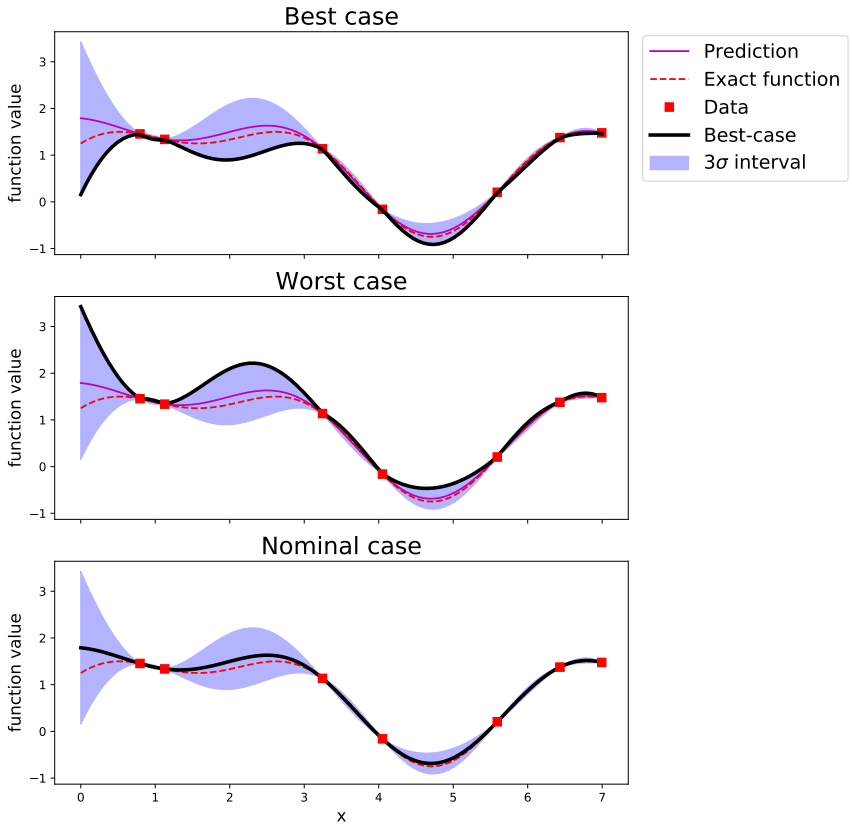


FIGURE 5.1: Example of Best-case, Worst-case and Nominal-case functions.

practice, by considering the minimum value of the Best-case and the Worst-case scenario, it is then possible to define an optimum range, in which the optimum of the modeled function is most likely to be, with a given confidence (proportional to the value of a). This is illustrated in Figure 5.2. Please note that the considered example is unconstrained. In the presence of constraints, the presented scenario would only be defined in the regions of the design space for which the EV of each constraint is lower than the given limit.

At the beginning of each SOMVSP iteration, the BC and the WC values are used in order to compare the predicted performance of each sub-problem and use this information in order to discard the less promising ones. More specifically, if the WC predicted feasible optimum of a given sub-problem yields a lower value than the BC predicted feasible optimum of a different sub-problem, it is expected that the latter is not worth further exploring as it will most likely not contain the global optimum. In this case, the non-optimal sub-problem is discarded and is not further explored for the rest of the optimization process. This is equivalent to determining whether the optimum range associated to two given sub-problems overlap or not. In practice, a

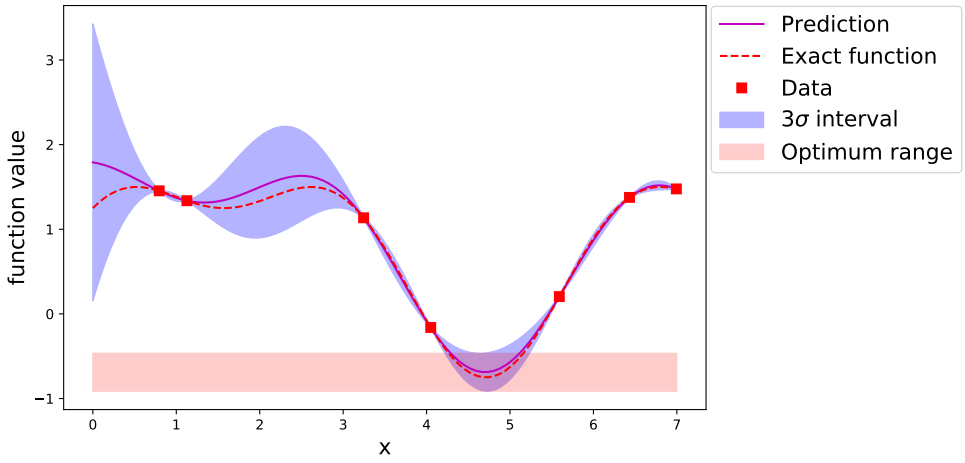


FIGURE 5.2: Example of a function optimum range, defined as the interval between the BC and the WC

sub-problem q is discarded if:

$$BC_q \geq WC_p \quad \text{for} \quad p = 1, ..., N_p \quad \text{and} \quad p \neq q \tag{5.7}$$

This condition must be evaluated for ever remaining sub-problem at the given iteration. For illustrative purposes, the comparison between 2 sub-problems is illustrated in Figure 5.3. As can be noticed, on the left side of the figure the variance associated to the two models is considerable, and the optimum ranges overlap. Therefore, in this case none of the two sub-problems would be discarded. Instead, the right size of the figure shows the comparison between the GPs obtained with a few more data samples per sub-problem. In this case the models are more accurate and the associated variance is smaller. As a consequence, it can be seen that the predicted optimum of sub-problem 1 yields a lower value than the one of sub-problem 2 with a given confidence level. In this case, the sub-problem 2 will be discarded as it is most likely that it does not contain the global optimum. Once the non-optimal sub-problems have been discarded (if present), the computational budget for the given SOMVSP iteration is allocated among the remaining sub-problems.

5.2.2 Computational budget allocation

At every SOMVSP iteration, the discarding of non-optimal sub-problems is followed by the allocation of a different computational budget to each remaining sub-problem. More specifically, at every iteration a budget of B_q data samples to be infilled is allocated to each remaining sub-problem q. In order to fairly compare sub-problems characterized by a potentially considerably different number of design variables, B_q is computed by taking into account both the predicted performance of a given sub-problem as well as its dimension, and it is defined as follows:

$$B_q = d_q \left(\frac{1 + \Delta_q}{2} \right) \tag{5.8}$$

where d_q is the total dimension of the sub-problem q (*i.e.*, sum of the continuous and discrete dimensions), while Δ_q is a term representing the relative performance of the considered sub-problem with respect to the remaining others. It is computed as:

$$\Delta_q = \frac{NC_{max} - NC_q}{NC_{max} - NC_{min}} \tag{5.9}$$

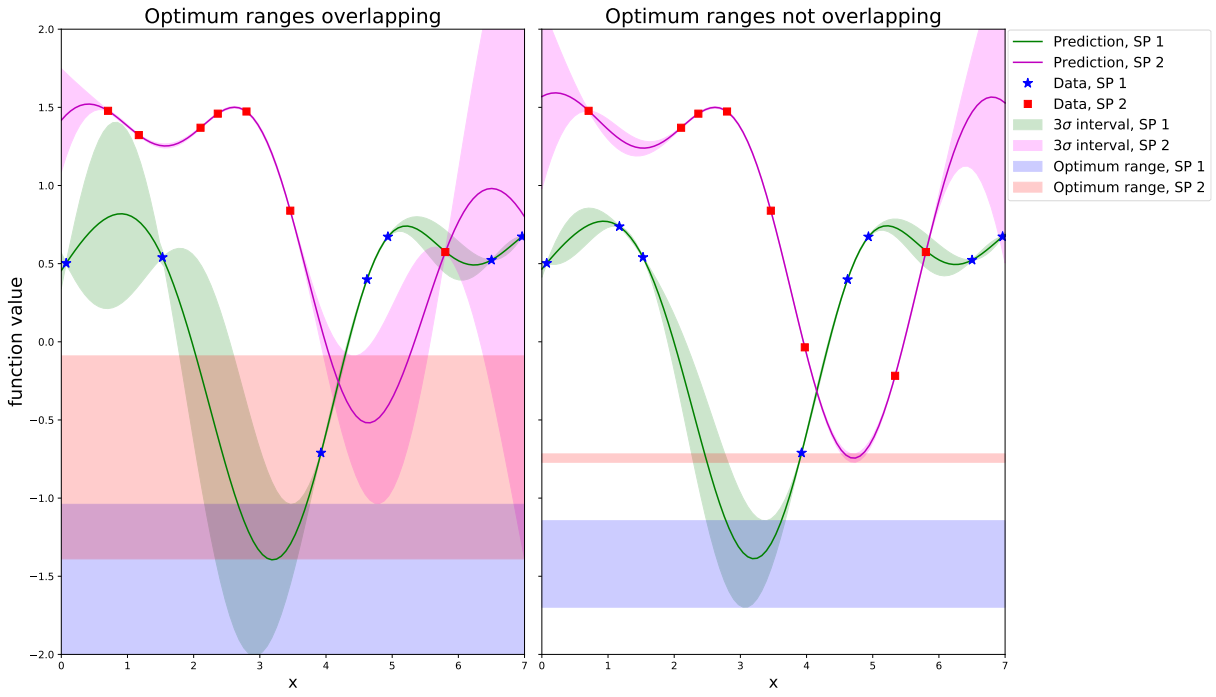


FIGURE 5.3: Example of comparison between optimum ranges between two Sub-Problems (SP) over different iterations. In the left figure, the optimum ranges of the two SP overlap and it is therefore not possible to accurately predict whether one of the two contains the global optimum. In the right figure, instead, the two optimum ranges do not overlap and it is therefore possible to discard the SP2 as it does most likely not contain the global optimum.

where NC_{max} and NC_{min} are respectively the largest and lowest NC values among the remaining sub-problems. As a result, a budget equal to half its total dimension is assigned to the least promising sub-problem, whereas a computational budget equal to its total dimension is assigned to the most promising one.

5.2.3 Bayesian optimization of remaining sub-problems

Following the computational budget allocation, each remaining sub-problem is independently optimized with the help of a mixed-variable BO similar to the one presented in Chapter 4, and by infilling a number of data samples proportional to the allocated budget. Due to the possibility of having different sub-problems characterized by a considerably different number of constraints, defining the BO acquisition function through the use of the Probability of Feasibility, as discussed in Chapter 4, might result in an uneven comparison between data samples as well as in uneven optimization performance between sub-problems. For this reason, the acquisition function considered when relying on the SOMVSP is the Expected Improvement under maximum Expected Violation constraints. The newly infilled data sample for each sub-problem is therefore defined as follows:

$$\begin{aligned} \{\mathbf{x}^n, \mathbf{z}^n\} &= \text{argmax}\left(EI(\mathbf{x}, \mathbf{z})\right) \\ \text{s.t.} \quad & EV_i(\mathbf{x}, \mathbf{z}) \leq t_i \quad \text{for} \quad i = 1, \dots, n_g \\ \text{w.r.t.} \quad & \mathbf{x} \in F_x, \ \mathbf{z} \in F_z \end{aligned} \tag{5.10}$$

Please note that in this case, $\mathbf{x}$ and $\mathbf{z}$ refer to continuous and discrete variables the considered sub-problem depends on. By consequence, the acquisition function optimization is performed in

a different search space for each considered sub-problem. Similarly, each infill process is only subject to the EV related to the constraints the considered sub-problem is subject to.

In Eq. 5.10 the values of t_i are constant. However, the possibility of reducing the values of t_i along the optimization process can also be considered. If properly tuned, this can allow to favor the exploration of the design space during the early stages of the optimization and subsequently favor the exploitation of the incumbent solution. Nevertheless, for clarity and synthesis purposes, the results obtained in this work are obtained by considering t_i to be constant along the optimization.

5.2.4 Algorithm overview

In the previous paragraphs, the main steps comprising an iteration of the proposed SOMVSP are detailed. By repeating this process, the number of remaining sub-problems is expected to gradually decrease, thus allowing to focus the computational budget on the most promising ones. In general, the proposed optimization strategy continues until a stopping criterion is reached. In this work, a predefined total computational budget (*i.e.,* number of function evaluations) is allocated to the optimization process, which performs SOMVSP iterations until the computational budget is depleted. This choice allows to analyze the performance of the proposed algorithm in a context of computationally intensive design problems. For illustrative purposes, a visual representation of the SOMVSP algorithm is provided in Figure 5.4.

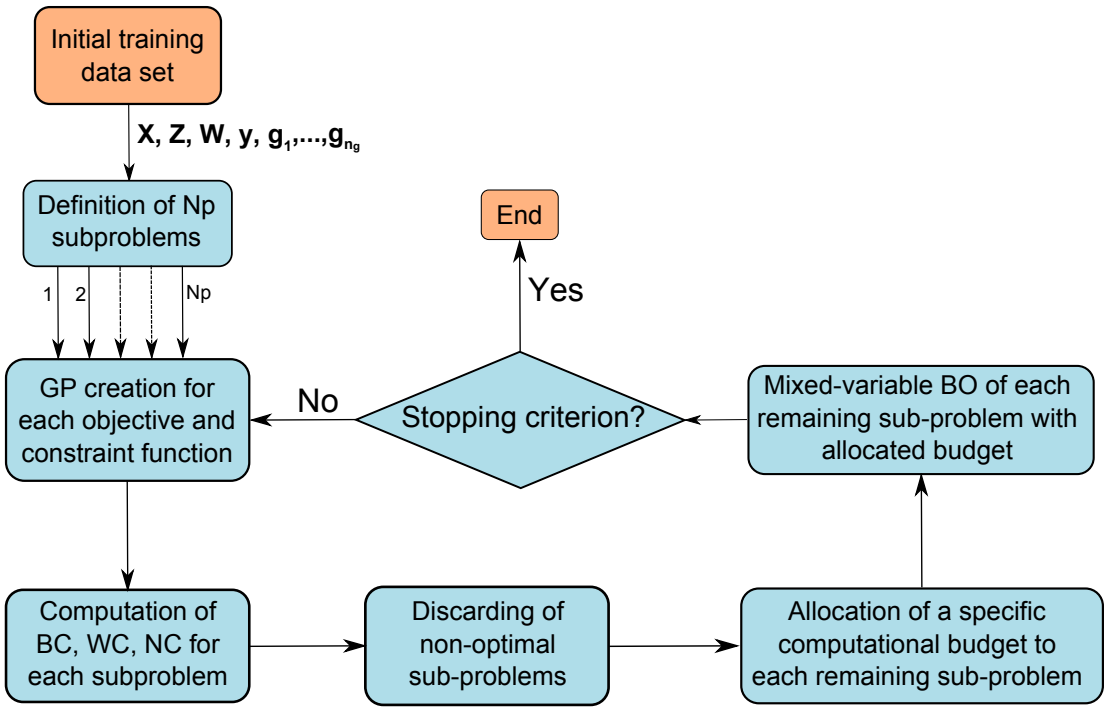


FIGURE 5.4: Budget allocation strategy for the optimization of variable-size design space problems.

The SOMVSP described in the previous paragraphs relies on the idea of using the information provided by the various sub-problems GP models in order to determine which ones are more likely to actually yield the global optimum, and allocating the computational budget accordingly. However, it might be important to note that the criteria which are used in order to determine how promising a given sub-problem is and how to allocate the computational budget among sub-problems are defined empirically. Therefore, these criteria might require to be adapted or redefined in order to better suit the characteristics of particular considered VSDSP, while still relying on the same working principle. The optimization results which can be obtained with the proposed SOMVSP on two different test-cases are shown in Section 5.4.

5.3 Variable-size design space kernel

The Budget allocation strategy for the optimization of variable-size design space problems presented in Section 5.2 provides a more efficient alternative than the separate and independent optimization of every fixed-sized sub-problem when dealing with VSDSP, as is shown in Section 5.4. However, for particularly complex and computationally costly problems characterized by a large dimensional variables combinatorial search space, the proposed method might still not be viable. For this reason, an alternative approach for the optimization of VSDSP is proposed in this Section. The underlying idea is to adapt the mixed-variable BO discussed in Chapter 4 for the solution of variable-size design space problems by defining a single Gaussian Process for each variable-size design space function characterizing a given problem (*i.e.,* objective and constraints). In order to do this, it is necessary to define a kernel allowing to characterize the covariance between data samples which belong to partially different design spaces, and therefore contain partially different sets of variables. In this Section, two alternative definitions for a variable-size design space kernel are discussed, and the adaptations required to optimize an infill criterion in a variable-size design space are commented. The first proposed approach relies on grouping the GP training data according to the sub-problem the samples belong to, in order to allow computing the covariance between different sub-problems. The second proposed approach, instead, relies on grouping the training data according to their dimensional variables values, thus providing a more complex but more accurate modeling of variable-size design space functions.

5.3.1 Sub-problem-wise decomposition kernel

As discussed in Section 5.2, a generic VSDSP can be decomposed into N_p sub-problems, each one characterized by a fixed-sized mixed variable search space. Furthermore, each sub-problem can be seen as a level of a single (scalar) dimensional variable w defined in the combinatorial space of $\mathbf{w}$, which characterizes the sub-problem the considered sample belongs to. A first possible way of defining a kernel in a variable-size design space consists in separately computing the within sub-problem covariance (*i.e.,* covariance between samples belonging to the same sub-problem) and the between sub-problems covariance (*i.e.,* covariance between data samples belonging to different sub-problems). By definition, the within sub-problem covariance can be represented through a mixed-variable kernel, as discussed in Chapter 3. The between sub-problems covariance, instead, can be represented as a discrete kernel defined on the search space of the (combinatorial) dimensional variable w. The global variable-size design space kernel can then be computed as the sum between the within sub-problem and between sub-problems covariances. This approach is referred to as Sub-Problem-Wise (SPW) decomposition in the remainder of this chapter.

Let q be one of the N_p levels of the dimensional variable w and let $\mathbf{x}_q$ and $\mathbf{z}_q$ be the continuous and discrete variables on which the objective and constraint functions associated to the sub-problem q depend. The first proposed variable-size search space kernel is defined as follows:

$$k\left((\mathbf{x},\mathbf{z},w),(\mathbf{x}',\mathbf{z}',w')\right) = \sum_{q=1}^{N_p} k_{x_q}(\mathbf{x}_q,\mathbf{x}'_q)\cdot k_{z_q}(\mathbf{z}_q,\mathbf{z}'_q)\cdot \delta_q(w,w') + k_w(w,w') \tag{5.11}$$

where $\delta_q(w,w')$ is a simil-Kronecker function which yields 1 in case both w and w' are equal to q, and 0 otherwise:

$$\delta_q(w,w') = \begin{cases} 1 & \text{if} \quad w = w' = q \\ 0 & \text{else} \end{cases} \tag{5.12}$$

This function can be seen as a kernel constructed in the following fashion:

$$\delta_q(w,w') = k_w(w,w') = \langle \phi_{\delta_q}(w), \phi_{\delta_q}(w')\rangle = \phi_{\delta_q}(w)\cdot\phi_{\delta_q}(w') \tag{5.13}$$

with ϕ_{δ_q} defined as a mapping function returning 1 for input values equal to q and 0 otherwise:

$$\phi_{\delta_q}(w) = \begin{cases} 1 & \text{if} \quad w = q \\ 0 & \text{else} \end{cases} \tag{5.14}$$

The kernel defined in Eq. 5.11 is defined through sums and products of one-dimensional kernels. By consequence, the validity of the global kernel can be ensured as long as each one-dimensional kernel is properly defined, which is already discussed in Chapter 3. The dimensional variable kernel k_w characterizing the covariance between the sub-problems, instead, only depends on the dimensional variable which is shared by all data samples. It can be constructed with the same logic as for discrete variables and, for clarity purposes, the same parameterizations as for the discrete variables are used for k_w in this work. However, in order to allow for a more refined modeling of the function trends, a heteroscedastic adaptation might be necessary. In practice, Eq. 5.11 is defined so that in case the two considered individuals belong to the same sub-problem, one of the N_p terms of the sum contributes to the covariance value while the remaining $N_p - 1$ yield 0. As a result, the total covariance value is computed as the sum between the within sub-problems covariance and the between sub-problem variance. If instead the two individuals do not belong to the same sub-problem, the covariance value is only computed as the between sub-problem covariance, as all the N_p terms of the sum are null.

Similarly to what is proposed by Roustant *et al.* [108] for the modeling of discrete variables with large number of levels, the global variable-size design space kernel can also be represented under the form of a symmetric block matrix,

$$A = \begin{bmatrix} W_1 + B_{1,1} & B_{1,2} & \dots & B_{1,N_p} \\ B_{2,1} & W_2 + B_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & B_{N_p-1,N_p} \\ B_{N_p,1} & \dots & B_{N_p,N_p-1} & W_{N_p} + B_{N_p,N_p} \end{bmatrix} \tag{5.15}$$

where W_q is the within sub-problem covariance matrix associated to the sub-problem q and computed as:

$$W_q = k_{x_q}(\mathbf{x}_q, \mathbf{x}'_q) \cdot k_{z_q}(\mathbf{z}_q, \mathbf{z}'_q) \tag{5.16}$$

while $B_{q,p}$ represents the covariance between the sub-problems q and p computed as:

$$B_{q,p} = k_w(w = q, w' = p) \tag{5.17}$$

The kernel described above allows to compute the covariance between data samples which belong to partially different search spaces by grouping the samples according to the sub-problem they belong to. However, this approach does not allow to exploit the information which may be provided by the fact that part of the design variables can be shared between data samples belonging to different sub-problems. In order to better exploit this additional information, an alternative variable-size design space kernel based on a decomposition by dimensional variable rather than by sub-problem is proposed in the following paragraphs.

5.3.2 Dimensional variable-wise decomposition

Without loss of generality, the generic VSDSP defined in Eq. 2.1 can formulated in such a way that each dimensional variable is associated to a specific number of continuous and discrete variables which may be active (or not) depending on its value. In the same way, it is also possible to formulate the problem in such a way that each continuous or discrete variable which does not always influence the problem function only depends on a single dimensional variable. In practice,

this translates into having a search space and a feasibility domain that directly depend on the dimensional variables levels, rather than on the dimensional variables category (*i.e.,* dimensional variable level combinations). The global VSDSP can therefore be decomposed into groups of design variables which depend on a given dimensional variable, plus the continuous and discrete variables which are shared among all the sub-problems. Within the framework of complex system design, this decomposition of the problem can be seen as the variable-size modeling of the different components characterizing a given system. For instance, a dimensional variable can characterize the type of engine to be installed on a launch vehicle stage. Depending on this choice, different engine design parameters can influence the system performance functions. Symmetrically, the engine design parameters only depend on the type of engine that is chosen, and is not influenced by other dimensional variables, such as the possible presence of lifting surfaces.

If this formulation of VSDSP is considered, an alternative variable-size design space kernel can be defined by considering a separate and independent kernel for each dimensional variable (*i.e.,* sub-system) and the continuous and discrete variables which depend on it. In fact, it can be noticed that each dimensional variable related to a number of possibly active continuous and discrete design variables depending on its value is equivalent to a VSDSP characterized by a single dimensional variable, and can therefore be modeled through the SPW decomposition kernel described in the previous paragraph. The second variable-size design space kernel proposed in this chapter can then be computed as a product of n_w SPW kernel (*i.e.,* one for each dimensional variable). This approach is referred to as Dimensional Variable-Wise (DVW) decomposition in the remainder of this chapter. Let $\mathbf{x}_{d_l}$ and $\mathbf{z}_{d_l}$ be the 'active' continuous and discrete variables associated to level l of the d_{th} dimensional variable (*i.e.,* $w_d = l$). Let $\mathbf{x}_s$ and $\mathbf{z}_s$ be the continuous and discrete variables which are 'shared' between all the samples and do not depend on a dimensional variable. The DVW decomposition kernel can be computed as follows:

$$k\left((\mathbf{x},\mathbf{z},\mathbf{w}),(\mathbf{x}',\mathbf{z}',\mathbf{w}')\right) = \prod_{d=1}^{n_w}\left(\sum_{l=1}^{l_{w_d}} k_{x_{d_l}}(\mathbf{x}_{d_l},\mathbf{x}'_{d_l})\cdot k_{z_{d_l}}(\mathbf{z}_{d_l},\mathbf{z}'_{d_l})\cdot\delta_l(w_d,w'_d) + k_{w_d}(w_d,w'_d)\right) k_{x_s}(\mathbf{x}_s,\mathbf{x}'_s)\cdot k_{z_s}(\mathbf{z}_s,\mathbf{z}'_s) \tag{5.18}$$

where l_{w_d} is the total number of levels characterizing the d_{th} dimensional variable w_d. In this case, each term of the product of Eq. 5.18 can be schematically represented by an $n_{w_d}\times n_{w_d}$ matrix A_d defined in the same way as the matrix presented in Eq. 5.15.

$$A_d = \begin{bmatrix} W_{d_1}+B_{d_{1,1}} & B_{d_{1,2}} & \dots & B_{d_{1,n_{w_d}}} \\ B_{d_{2,1}} & W_{d_2}+B_{d_{2,2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & B_{d_{n_{w_d}-1,n_{w_d}}} \\ B_{d_{n_{w_d},1}} & \dots & B_{d_{n_{w_d},n_{w_d}-1}} & W_{d_{n_{w_d}}}+B_{d_{n_{w_d},n_{w_d}}} \end{bmatrix} \tag{5.19}$$

The DVW decomposition kernel offers a more accurate modeling of VSDSP with respect to the SPW alternative as it allows to exploit the information which can be provided by design variables shared between data samples belonging to different sub-problems (*i.e.,* variables shared between different levels of a given dimensional variables). However, this kernel definition is also more complex and is usually characterized by a larger number of hyperparameters. As a consequence, the training of the GP model might be more difficult in case an insufficient amount of training data is provided.

5.3.3 Variable-size design space Gaussian Process training

Although the two alternative kernels defined in the previous paragraph allow to compute the covariance between samples defined in a variable-size design space, they are computed through sums and products of the purely continuous and purely discrete kernels discussed in Chapter 3. As a consequence, the training of the proposed variable-size design space GP only requires to optimize the marginal likelihood in a fixed-size continuous search space. By consequence, also for variable-size design space GP the training is performed with the help of a L-BFGS-B [24] algorithm in the same fashion as is discussed in Chapter 3.

5.3.4 Infill criterion optimization

In order to perform the BO of VSDSP, it is necessary to define and optimize an acquisition function within the variable-size design space allowing to determine the most promising locations of this search space. In this chapter, the data sample $\{\mathbf{x}^n, \mathbf{z}^n, \mathbf{w}^n\}$ to be infilled at each iteration of the proposed VSDSP BO is computed by relying on the following infill criterion:

$$\begin{aligned} \mathbf{x}^n, \mathbf{z}^n, \mathbf{w}^n \quad = \quad & \text{argmax}(EI(\mathbf{x}, \mathbf{z}, \mathbf{w})) \\ & \text{s.t.} \quad EV\left(g_c(\mathbf{x}, \mathbf{z}, \mathbf{w}_q)\right) \leq t_c \qquad \text{for} \qquad c = 1, ..., n_g(\mathbf{w}) \\ & \text{w.r.t.} \quad \mathbf{x} \in F_x(\mathbf{w}) \subseteq \mathbb{R}^{n_x(\mathbf{w})} \\ & \qquad\quad \mathbf{z} \in \prod_{d=1}^{n_z(\mathbf{w})} F_{z_d} \\ & \qquad\quad \mathbf{w} \in F_w \end{aligned} \tag{5.20}$$

Similarly to the SOMVSP proposed in Section 5.2, the PoF used in Chapter 4 is not applicable in this case, as it would penalize the candidate solutions which are subject to a larger number of constraints. For this reason, the EV criterion is used in order to take into account the presence of constraints. It can be noticed that the optimization problem defined in Eq. 5.20 is also defined in the variable-size design space, and it is therefore necessary to solve an auxiliary VSDSP optimization. However, the objective and constraint functions of this optimization auxiliary problem present a negligible computational cost when compared to the actual problem. By consequence, the infill criterion optimization problem can be solved by relying on heuristic VSDSP optimization algorithms, such as the GA variant proposed in [93]. However, due to the fact that the implementation of these heuristic VSDSP algorithm requires problem specific inputs and parameterization, the results presented in this chapter are obtained by optimizing a separate acquisition function for each sub-problem, and by selecting the data sample which yields the best acquisition function value among all the sub-problems:

$$\{\mathbf{x}^n, \mathbf{z}^n, \mathbf{w}^n\} = \text{argmax} \left\{ \begin{array}{lll} \text{argmax} & \left(EI(\mathbf{x}, \mathbf{z}, \mathbf{w}_q)\right) & \\ \text{s.t.} & EV\left(g_c(\mathbf{x}, \mathbf{z}, \mathbf{w}_q)\right) < t_c \qquad \text{for} & c = 1, ..., n_g(\mathbf{w}_q) \\ \text{w.r.t.} & \mathbf{x} \in F_x(\mathbf{w}_q) \subseteq \mathbb{R}^{n_x(\mathbf{w}_q)} & \\ & \mathbf{z} \in \prod_{d=1}^{n_z(\mathbf{w}_q)} F_{z_d} & \end{array} \right\} \tag{5.21}$$
$$\text{for} \quad q = 1, \ldots, N_p$$

The N_p optimization problems defined by Eq. 5.21 are standard mixed-variable optimization problems. By consequence, similarly to the SOMVSP discussed in Section 5.2, the results presented in this article are obtained by optimizing the infill criterion with the help of a mixed

continuous/discrete GA similar to the one used in Chapter 3, with the exception that a constraint dominance mechanism is included in the selection criterion in order to ensure the converge towards feasible solutions (*i.e.,* solutions for which all the EV values are lower than the given threshold).

5.4 Applications and Results

In order to assess and compare the performance of the two proposed BO-based approaches (*i.e.,* SOMVSP and variable-size design space kernel based BO) for the optimization of VSDSP, two test-cases with different characteristics are considered. More specifically, an analytical test-case and an aerospace engineering design problem are optimized. The proposed methods are applied to each benchmark with different discrete kernel parameterizations as well as different parameters (*i.e.,* SPW and DVW approaches for the variable-size design space kernel and different values of a for the SOMVSP). Please note that for clarity purposes, the SOMVSP results are referred to with the acronym BA (Budget Allocation). The objective is to assess the relative performance between the two proposed approaches as well as the impact of the considered algorithm parameters on the resulting convergence speed. The 2 discrete kernel parameterizations which are considered for these benchmarks are the Compound Symmetry (CS) and the Latent Variables (LV), due to their robustness with respect to the characteristics of the modeled functions as well as the scaling of the number of hyperparameters as a function of the problem discrete search space size. Because of the implementation limitations and poor performance of the few existing heuristic algorithms allowing to solve VSDSP, the reference method which is considered for the presented benchmarks is defined as a separate and independent mixed-variable BO of each sub-problem characterizing the considered VSDSP (referred to as IO). The mixed-variable BO is very similar to the algorithm presented in Chapter 4, with the main difference being that the acquisition function is defined as the EI under EV constraints rather than as the product between the EI and the PoF. The total computational budget allocated for each given test-case is distributed among the sub-problems proportionally to their total dimension. The initial data set which is provided to compare optimization algorithms is sampled on the global design space, *i.e.,* on every continuous and discrete design variable the considered problem depends on. The continuous variables are sampled through a single continuous LHS [86], while the discrete variables are drawn from a uniform discrete distribution. Finally, these data samples are randomly associated to each sub-problem (or to a dimensional variable category). The number of samples allocated to each sub-problem is proportional to its total dimension (*i.e.,* sum of continuous and discrete dimensions). In order to quantify and compensate the influence of the initial DoE random nature, each optimization problem is solved multiple times with different initial training data sets. The actual number of repetitions depends on the optimization problem which is being considered.

5.4.1 Benchmark analysis

In the following paragraphs, the 2 proposed alternative solutions for the optimization of VSDSP are tested on a number of test-cases with different kernel parameterizations and different parameters. More specifically, an analytical problem and an aerospace engineering design test-case are considered. These benchmarks present different characteristics in terms of number of sub-problems, combinatorial complexity of the dimensional variable design space, as well as in terms of sub-problem design space dimensions and sub-problem specific constraints. The main properties of the considered test-cases as well as the simulation details are provided below:

Variable-size design space Goldstein function

- 5 continuous variables, 4 discrete variables, 2 dimensional variables
- 8 sub-problems

- 648 equivalent continuous problems
- 1 constraint
- Initial data set size: 104 samples (*i.e.,* 2 samples per dimension of each sub-problem)
- Number of infilled samples: 104
- Compared methods:

  - Independent mixed-variable BO of each sub-problem (IO) with both CS and LV kernels

  - SOMVSP (BA) with CS and LV kernels and values of a of 2 and 3

  - Variable-size design space kernel BO with CS and LV kernels with both SPW and DVW approaches
- Acquisition function: EI under EV constraints

Multi-stage launch vehicle design

- 18 continuous variables, 14 discrete variables, 3 dimensional variables
- 6 sub-problems
- 29136 equivalent continuous problems
- 19 constraint
- Initial data set size: 122 samples (*i.e.,* 1.5 sample per dimension of each sub-problem)
- Number of infilled samples: 58
- Compared methods:

  - Independent mixed-variable BO of each sub-problem (IO) with both CS and LV kernels

  - SOMVSP (BA) with CS and LV kernels and values of a of 3

  - Variable-size design space kernel BO with CS and LV kernels with both SPW and DVW approaches
- Acquisition function: EI under EV constraints
- Large number of continuous and discrete design variables. Large number of constraints.

Implementation

Similarly to the modeling performance benchmark analysis of Chapter 3, the results presented in the following paragraphs are obtained with the following implementation. The optimization routine overhead is written in Python 3.6. The GP models are created with the help of GPflow [83], a Python based toolbox for GP-based modeling relying on the Tensorflow framework [1] (version 1.13). The surrogate model training is performed with the help of a Bounded Limited memory Broyden - Fletcher - Goldfarb – Shanno (L-BFGS-B) algorithm [24], whereas the acquisition functions are optimized the help of a constraint domination based mixed continuous/discrete Genetic Algorithm (GA) [121] implemented by relying on the Python based toolbox DEAP [43].

5.4.2 Variable-size design space Goldstein function

The first analytical test-case that is considered in this Chapter is a modified variable-size design space version of the Goldstein function considered for modeling purposes in Chapter 3 and constrained optimization purposes in Chapter 4. The global VSDSP is characterized by 5 continuous variables, 4 discrete variables and 2 dimensional variables. Depending on the dimensional variable values, 8 different sub-problems can be identified, with total dimensions of 6 or 7, ranging from 2 continuous variables and 4 discrete variables to 5 continuous variables and 2 discrete variables. All of the sub-problems are subject to a variable-size design space constraint. The resulting optimization problem can be defined as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}) \\ \text{w.r.t.} \quad & \mathbf{x} = \{x_1,\dots,x_5\} \text{ with } x_i \in [0,100] \text{ for } i = 1,5 \\ & \mathbf{z} = \{z_1,\dots,z_4\} \text{ with } z_i \in \{0,1,2\} \text{ for } i = 1,4 \\ & \mathbf{w} = \{w_1,w_2\} \text{ with } w_1 \in \{0,1,2,3\} \text{ and } w_2 \in \{0,1\} \\ \text{s.t.:} \quad & g(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0 \end{aligned} \tag{5.22}$$

where:

$$f(\mathbf{x},\mathbf{z},\mathbf{w}) = \begin{cases} f_1(x_1,x_2,z_1,z_2,z_3,z_4) & \text{if } w_1 = 0 \text{ and } w_2 = 0 \\ f_2(x_1,x_2,x_3,z_2,z_3,z_4) & \text{if } w_1 = 1 \text{ and } w_2 = 0 \\ f_3(x_1,x_2,x_4,z_1,z_3,z_4) & \text{if } w_1 = 2 \text{ and } w_2 = 0 \\ f_4(x_1,x_2,x_3,x_4,z_3,z_4) & \text{if } w_1 = 3 \text{ and } w_2 = 0 \\ f_5(x_1,x_2,x_5,z_1,z_2,z_3,z_4) & \text{if } w_1 = 0 \text{ and } w_2 = 1 \\ f_6(x_1,x_2,x_3,x_5,z_2,z_3,z_4) & \text{if } w_1 = 1 \text{ and } w_2 = 1 \\ f_7(x_1,x_2,x_4,x_5,z_1,z_3,z_4) & \text{if } w_1 = 2 \text{ and } w_2 = 1 \\ f_8(x_1,x_2,x_3,x_5,x_4,z_3,z_4) & \text{if } w_1 = 3 \text{ and } w_2 = 1 \end{cases} \tag{5.23}$$

and:

$$g(\mathbf{x},\mathbf{z},\mathbf{w}) = \begin{cases} g_1(x_1,x_2,z_1,z_2) & \text{if } w_1 = 0 \\ g_2(x_1,x_2,z_2) & \text{if } w_1 = 1 \\ g_3(x_1,x_2,z_1) & \text{if } w_1 = 2 \\ g_4(x_1,x_2,z_3,z_4) & \text{if } w_1 = 3 \end{cases} \tag{5.24}$$

For clarity purposes, the analytical definitions of $f_1(\cdot),\dots,f_8(\cdot)$ and $g_1(\cdot),\dots,g_4(\cdot)$ are not provided in this chapter but can be found in Appendix C. A synthesis of the characteristics of the various sub-problems comprising the considered variable-size design space function problem is presented in Table 5.1. Furthermore, the value ranges of the feasible objective function for each sub-problem is shown in Figure 5.5. It can be seen that the feasible values of the different sub-problems overlap over a large part of their objective function range, thus making the identification of the optimal sub-problem challenging.

The compared algorithms are initialized with a total data set of 104 data samples, which is equivalent to providing 2 samples for every dimension of each of the 8 considered sub-problems. Subsequently, the optimizations are performed by relying on 104 additional function evaluations. The results obtained over 10 repetitions are provided in Figures 5.6, 5.7, 5.8 and 5.9
Overall, the results show that all the variants of the proposed algorithms, namely the SOMVSP and the variable-size design space kernel based BO algorithm, provide a faster and more consistent convergence towards the considered VSDSP optimum when compared to the independent

Sub-problem	SP 1	SP 2	SP 3	SP 4	SP 5	SP 6	SP 7	SP 8
N° continuous variables	2	3	3	4	3	4	4	5
N° discrete variables	4	3	3	2	4	3	3	2
N° constraints	1	1	1	1	1	1	1	1

Global VSDSP	
N° continuous variables	5
N° discrete variables	4
N° dimensional variables	2
N° discrete categories	648
N° constraint	1

TABLE 5.1: Defining characteristics of the sub-problems comprising the variable-size design space Goldstein function optimization problem.

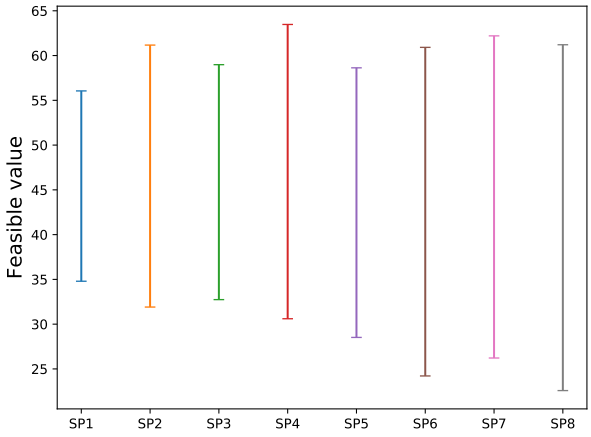


FIGURE 5.5: Value range of the feasible objective function for each sub-problem of the variable-size design space Goldstein function.

optimization of each sub-problem. For the SOMVSP, this difference can be explained by the fact that the proposed strategy allows to better focus the computational budget towards the most promising sub-problems through budget allocation and sub-problem discarding. By consequence, very few function evaluations are wasted onto the least promising sub-problems. For the variable-size design space kernel based BO algorithm, instead, the better performance can be explained by the fact that the GP models are created and trained by relying on the entirety of the available data (*i.e.,* the data set in the variable-size design space) and can therefore better exploit the available information. On the other hand, the IO of each sub-problem relies on GP which are created by relying only on data sets specific to each sub-problem, and rely therefore on a lower amount of information. As a consequence, the resulting modeling performance of the problem functions is expected to be less accurate.

When considering the relative performance between the 2 proposed methods (and their variants), the results show a better convergence of the variable-size design space kernel based BO algorithm with respect to the SOMVSP, in terms of both convergence rate, as can be seen in Figure 5.6, as well as optimum value at convergence, as is shown in Figure 5.7. As previously mentioned, this can be explained by the fact that the variable-size design space BO can rely on the entirety of the available data, whereas the SOMVSP only performs optimizations with respect to sub-problem specific data sets. Therefore, even in the case in which the SOMVSP has

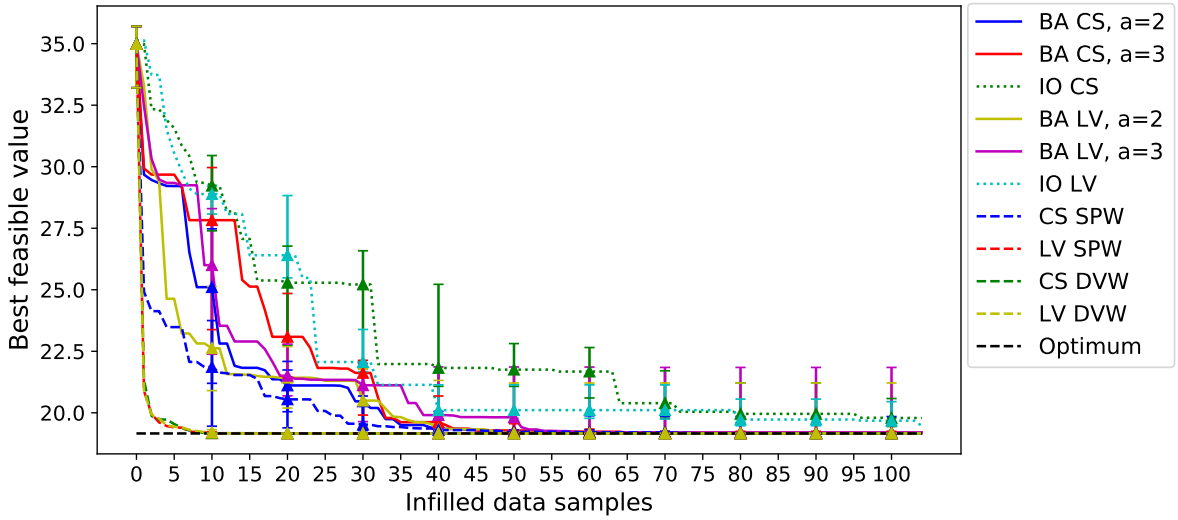


FIGURE 5.6: Comparison of the convergence rate of various VSDSP optimization algorithms during the BO of the mixed-variable Goldstein function over 10 repetitions. The continuous lines represent the SOMVSP alternatives, the dashed lines represent the variable-size design space kernel BO alternatives and the dotted lines represent the independent optimization of each sub-problem.

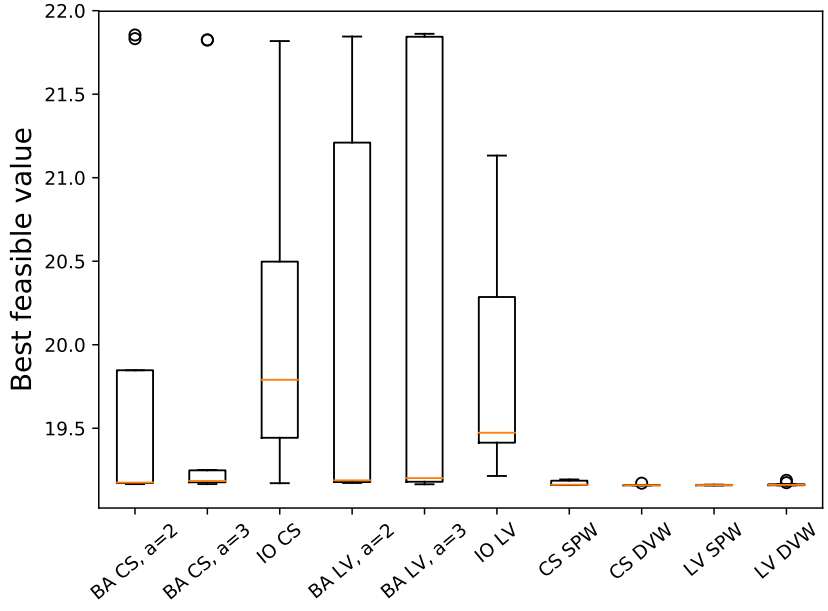


FIGURE 5.7: Comparison of the convergence value of different VSDSP optimization algorithms on the variable-size design space Goldstein function over 10 repetitions.

identified the optimal sub-problem and discarded all the others, the variable-size design space BO may still converge faster thanks to the information it can extrapolate from the data samples belonging to non-optimal sub-problems.

When analyzing the results provided by the different SOMVSP variants, a difference in convergence rate can be noticed for different values of a (but identical kernels). Indeed, the optimizations performed with a value of $a = 2$ provide a faster convergence when compared to the ones obtained with a value of $a = 3$. This is related to the fact that lower values of a result in a more frequent discarding of sub-problems, which can then result in a better allocation of the computational

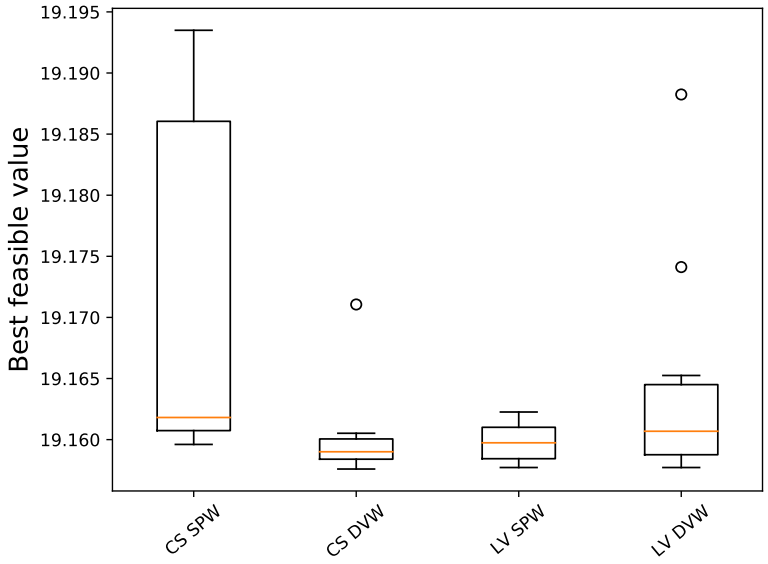


FIGURE 5.8: Comparison of the convergence value of different VSDSP optimization algorithms on the variable-size design space Goldstein function over 10 repetitions. Focus on the variable-size design space kernel based BO algorithms.

resources. However, lower values of a can also cause the proposed algorithm to discard the optimal sub-problems during the early stages of the optimization, when insufficient data is provided in order to make accurate choices. This phenomenon can, for instance, be noticed in the larger variance of the results obtained with the CS kernel and a value of $a = 2$ when compared to the ones obtained with the CS kernel and a value of $a = 3$. In order to better illustrate the effect of considering different values of a, the evolution of the remaining number of sub-problems along the optimization for different kernels is shown in Figure 5.9. As previously discussed, it can be seen that lower values of a result in a faster discarding of sub-problems (if the same kernel is considered). Furthermore, it can also be noticed that relying on the LV kernel tends to result in a faster discarding of problems when compared to the CS, which is related to the different accuracy of the model of error, as is mentioned in Chapter 3. Finally, the figure also shows that from half of the optimization onward, all of the compared SOMVSP variants tend to have discarded all the problems but one.

When analyzing the results provided by the variable-size design space kernel based BO variants, the first noticeable result is a considerably faster convergence of the DVW variant when compared to the SPW one. Indeed, the DVW variant consistently converges around the 10th infilled point, whereas the SPW variant requires more or less half of the allocated budget in order to yield the same performance. This difference is related to the fact that the DVW kernel is defined in such a way that it can rely on the variables shared between the different sub-problems in order to better model the considered function, while the SPW kernel only computes the covariance between samples defined in different sub-problems through their dimensional variable values. This difference becomes more noticeable when the CS discrete kernel is considered, due to the fact that it relies on a single covariance value between all of the sub-problems. However, note that this is not true for the DVW kernel, as in this case the covariance between sub-problems is computed as the product between the covariances between the values of the two dimensional variables. Finally, no significant difference of performance between the variable-size design space kernel based BO variants can be noticed from the values at convergence shown in Figure 5.8, with the exception of a slightly larger variance for the CS based SPW kernel. This can be explained by the fact that all of these compared methods are provided with enough function evaluations in

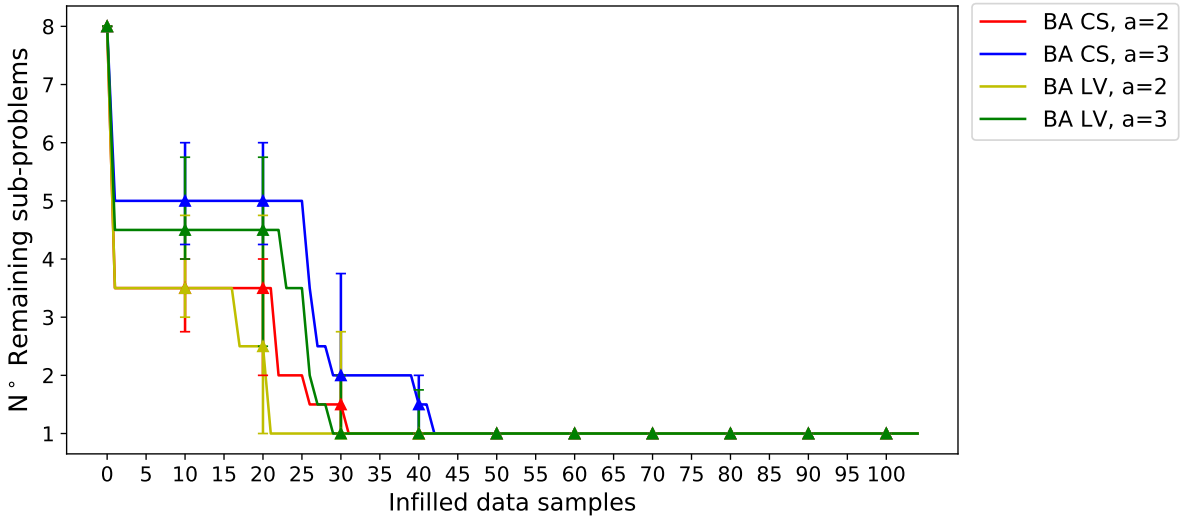


FIGURE 5.9: Comparison of the number of remaining sub-problems along the optimization process for different parameterizations of the SOMVSP on the variable-size design space Goldstein function over 10 repetitions.

order to properly refine the incumbent solution.

In order to better compare the performance of the different variable-size design space kernel based BO variants, the optimizations presented in the paragraph above are repeated by providing a smaller initial data set (*i.e.,* 52 data samples, 1 per dimension of each sub-problem) and a lower number of additional function evaluations (*i.e.,* 52 data samples). The results obtained over 10 repetitions are provided in Figures 5.10 and 5.11.

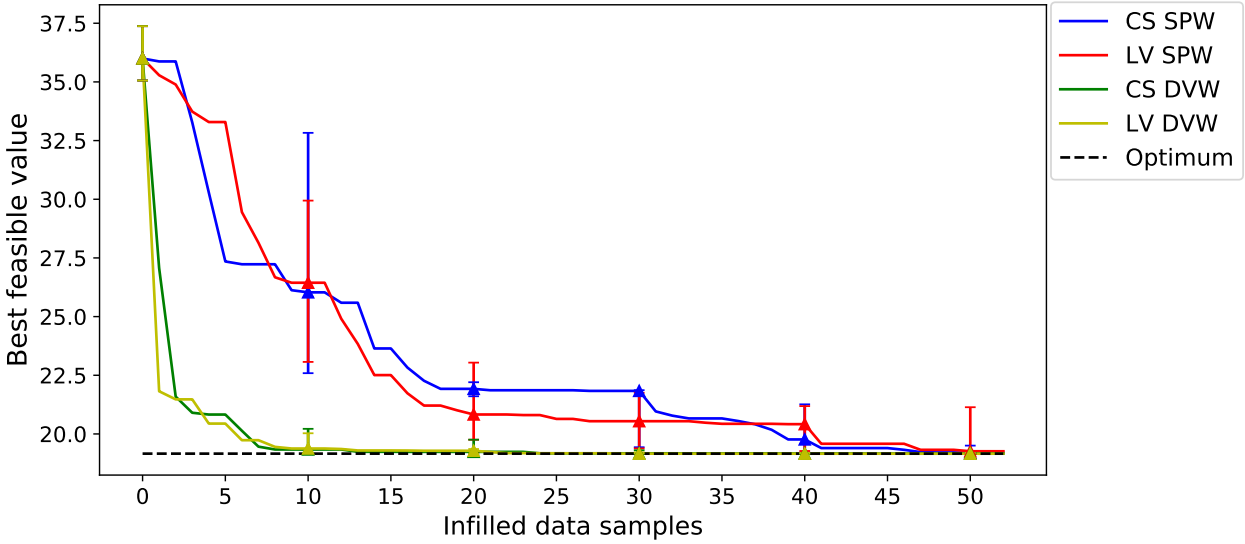


FIGURE 5.10: Comparison of the convergence rate of different VSDSP optimization algorithms during the BO of the mixed-variable Goldstein function over 10 repetitions. Focus on the the variable-size design space kernel BO alternatives.

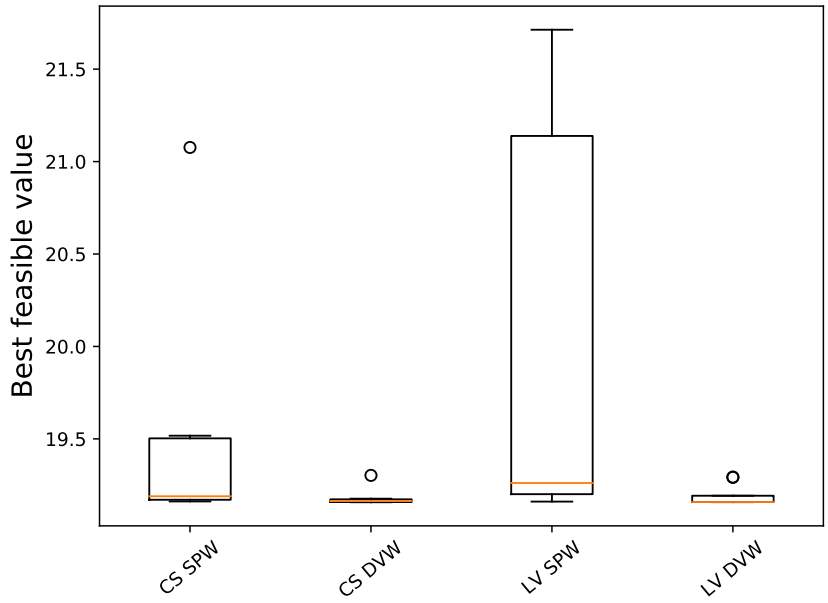


FIGURE 5.11: Comparison of the convergence value of different VSDSP optimization algorithms on the variable-size design space Goldstein function over 10 repetitions. Focus on the the variable-size design space kernel BO alternatives.

As can be expected, the convergence rate obtained in this case is slower if compared to the previously considered optimizations. This can be explained by the fact that smaller initial data sets usually result in GP models characterized by a lower modeling accuracy. As a consequence, a larger number of function evaluations is necessary in order to sufficiently refine the models before being able to identify the global optimum neighborhood. Furthermore, the obtained results also show that the relative performance between the DVW and the SPW kernels remains the same when considering smaller initial data sets, as the DVW kernel provides a considerably faster convergence rate. As for the previous optimizations, this is due to the fact that the DVW kernel can rely on a larger amount of information in order to compute the covariance between data samples characterized by different dimensional variable values.

5.4.3 Multi-stage launch vehicle architecture optimization

The second variable-size design space problem which is considered in this chapter is the preliminary optimization of a multi-stage launch vehicle architecture. This design problem requires to simultaneously determine the optimal number of stages characterizing the system (*i.e.,* 2 or 3) and determine the most suitable type of propulsion (*i.e.,* solid or liquid) for each stage. Given that each propulsive alternative is characterized by different continuous and discrete design variables as well as different constraints, the resulting optimization problem presents a variable-size design space. The objective function allowing to assess the performance of the system is defined as the Gross Lift-Off Weight (GLOW), which is computed as the sum of the payload mass M_{PL}, as well as the dry mass (M_d) and propellant mass (M_{prop}) of each stage:

$$GLOW = M_{PL} + \sum_{i=1}^{n_{stages}} \left(M_{d_i} + M_{prop_i} \right) \tag{5.25}$$

The target mission for which the launch vehicle is designed is the injection of a 500 kg payload into an 800 km Low Earth Orbit (LEO) [130]. A velocity increment ΔV of 7500 m/s is required in order to reach the considered LEO. However, due to gravity losses, the actual required velocity

increment is increased by a fixed margin. Given that a multi-stage architecture is considered, the total velocity increment ΔV is computed as the sum of the velocity increments provided by all of the stages:

$$\Delta V = \sum_{i=1}^{n_{stages}} \Delta V_i \tag{5.26}$$

Similarly to the launch vehicle propulsion performance optimization problem described in Chapter 4, the ΔV_i associated to a given stage is computed through the Tsiolkovsky rocket equation:

$$\Delta V_j = g_0 I_{sp} \ln\left(\frac{M_{i_j}}{M_{f_j}}\right) = g_0 I_{sp} \ln\left(\frac{M_{PL} + \sum_{j=i}^{n_{stages}} (M_{d_j} + M_{prop_j})}{\sum_{j=i+1}^{n_{stages}} (M_{d_j} + M_{prop_j}) + M_{d_i}}\right) \tag{5.27}$$

The specific impulse I_{sp} as well as the initial and final masses are computed differently depending on whether solid or liquid propulsion is considered. Furthermore, in order to ensure the feasibility of the proposed design, each stage must be characterized by a thrust-to-weight ratio (TW) larger than 1 in order to obtain a positive acceleration. In other words, each stage must provide a thrust sufficient to lift the total mass comprising both the considered stages and the ones above:

$$TW_i > 1 \rightarrow T_i > g\left(M_{PL} + \sum_{j=i}^{n_{stages}} (M_{d_j} + M_{prop_j})\right) \tag{5.28}$$

5.4.3.1 Liquid propulsion

Each liquid propulsion stage is characterized by 2 continuous design variables, namely the stage specific propellant mass M_{prop} and the thrust value T, as well as one discrete design variable characterizing the type of engine $Type_{eng}$ to be included in the design. Additionally, if the considered liquid propulsion stage is also the first stage of the launch vehicle, an additional discrete design variable representing the number of engines N_e must be considered. The continuous and discrete design variables characterizing each liquid propulsion stage are detailed in Table 5.2. Note that the continuous variable bounds vary as a function of the position of the considered architecture as well as on the overall system architecture.

Variable	Nature	Levels
T - Engine thrust [kN]	continuous	[-]
M_{prop} – Propellant mass [kg]	continuous	[-]
$Type_{eng}$ - Type of engine	discrete	type 1, type 2, type 3
N_e - Number of engines	discrete	1,2

TABLE 5.2: Variables characterizing each liquid propulsion stage

A schematic representation of the dependencies between the different disciplines characterizing the liquid propulsion stage as well as the continuous and discrete design variables they depend on is provided in Figure 5.12

5.4.3.2 Solid propulsion

The solid propulsion module is similar to the one used for the launch vehicle propulsion performance optimization problem described in Chapter 4. Each solid propulsion stage is characterized as a function of 4 continuous design variables, namely the nozzle throat diameter D_t, the nozzle exit diameter D_e, the combustion chamber pressure P_{comb} and the propellant mass M_{prop}. Additionally, the solid propulsion stage also depends on three discrete design variables: the type of propellant $Type_{prop}$, the type of material $Type_{mat}$ and the type of engine$Type_{eng}$. Furthermore, if the first stage is considered, a discrete variable N_b characterizing the number of boosters attached

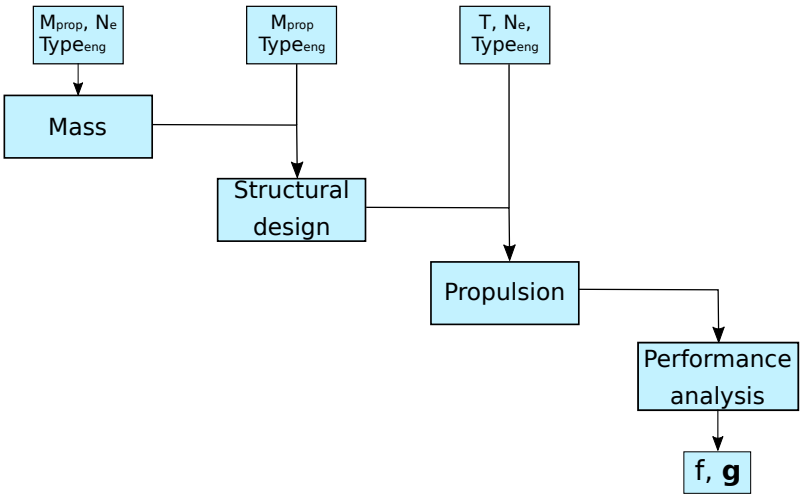


FIGURE 5.12: Schematic MDO representation of the liquid propulsion stage.

to the stage is also taken into account. The continuous and discrete design variables characterizing each solid propulsion stage are detailed in Table 5.3. Note that the continuous variable bounds vary as a function of the position of the considered architecture as well as on the overall system architecture.

Variable	Nature	Levels
D_t - Nozzle throat diameter [m]	continuous	[-]
D_e - Nozzle exit diameter [m]	continuous	[-]
P_{comb} – Chamber pressure [bar]	continuous	[-]
M_{prop} – Propellant mass [kg]	continuous	[-]
$Type_{prop}$ - Type of propellant	discrete	Butalite, Butalane, Nitramite, pAIM-120
$Type_{mat}$ - Type of material	discrete	Aluminium, Steel
$Type_{eng}$ - Type of engine	discrete	type 1, type 2, type 3
N_b - Number of booster	discrete	1,...,8

TABLE 5.3: Variables characterizing each solid propulsion stage

A schematic representation of the dependencies between the different disciplines characterizing the solid propulsion stage as well as the continuous and discrete design variables they depend on is provided in Figure 5.13

5.4.3.3 Variable-size design space problem formulation

The multi-stage launch vehicle architecture optimization described above can be formulated under the form of a variable-size design space problem by considering 3 dimensional variables, each one representing the type of propulsion to be included in one of the 3 considered stages. In order to simplify the problem, the unfeasible configurations are not taken into account, thus reducing the total number of sub-problems from 12 to 6. The remaining configurations are schematically represented in Figure 5.14.

The resulting dimensional variables w_1, w_2, w_3 are respectively characterized by 2, 2 and 3 levels. w_1 and w_2 have the purpose of determining whether the first and second stage are characterized by liquid or solid propulsion, whereas w_3 determines whether the first stage is characterized by solid propulsion, liquid propulsion (*i.e.,* 3 stage architecture) or whether it is not included in the architecture (*i.e.,* 2 stage architecture).

The resulting VSDSP is characterized by a total of 18 continuous variables, 14 discrete variables and 3 dimensional variables, thus resulting in 29136 discrete categories. Furthermore, the

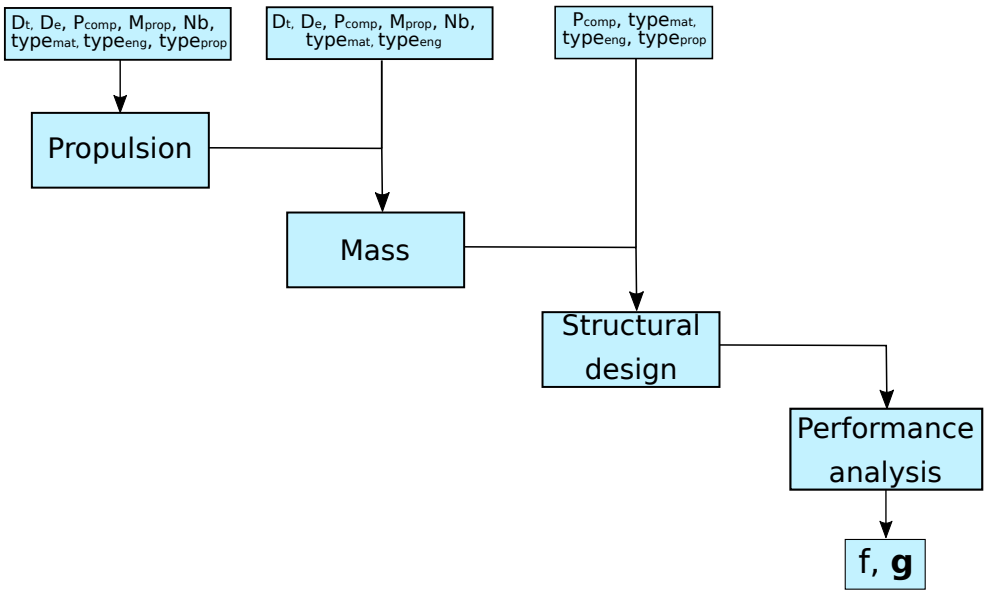


FIGURE 5.13: Schematic MDO representation of the solid propulsion stage.

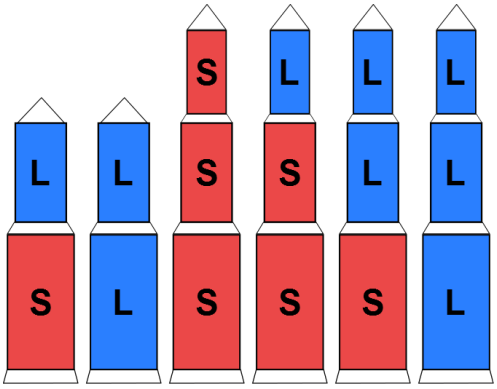


FIGURE 5.14: Considered launch vehicle architectures (S: Solid propulsion, L: Liquid propulsion).

problem problem is subject to 19 constraints. It can be formulated as follows:

$$
\begin{aligned}
\min \quad & GLOW(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \\
\text{w.r.t.} \quad & S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3 \\
\text{s.t.:} \quad & g_{\Delta V}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \\
& g_{TW_i}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \quad \text{for i } = 1,2,3 \\
& g_{g1_i}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \quad \text{for i } = 1,2,3 \\
& g_{g2_i}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \quad \text{for i } = 1,2,3 \\
& g_{g3_i}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \quad \text{for i } = 1,2,3 \\
& g_{g4i}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \quad \text{for i } = 1,2,3 \\
& g_{e_i}(S_1, L_1, S_2, L_2, S_3, L_3, w_1, w_2, w_3) \leq 0 \quad \text{for i } = 1,2,3
\end{aligned} \tag{5.29}
$$

where S_n and L_n respectively represent the design variables associated to the n-th solid and liquid propulsion stages, respectively. It is important to note that some constraints, such as the thrust-to-weight ratio (g_{TW}) and the geometrical (g_g) constraints, must be separately considered for each stage that is included in the launch vehicle architecture. A synthesis of the characteristics of the six sub-problems comprising the considered multi-stage launch vehicle architecture design

problem is presented in Table 5.4. Furthermore, the value range of the feasible objective function for each sub-problem is shown in Figure 5.15. Differently than for the variable-size design space Goldstein function, not all the feasible objective function ranges of the different sub-problems overlap. For instance, the best feasible performance value of the triple solid propulsion stage (SSS) is considerably larger than any feasible performance of the triple liquid propulsion stage (LLL). It is expected that the optimization algorithms should mostly explore the areas of the design space associated to the sub-problems SL, LL and LLL.

Sub-problem	SL	LL	SSS	SSL	SLL	LLL
N° continuous variables	6	4	12	10	8	6
N° discrete variables	5	3	10	8	6	4
N° discrete categories	48	32	27648	1152	192	64
N° constraints	8	3	19	14	9	4

Global VSDSP	
N° continuous variables	18
N° discrete variables	14
N° dimensional variables	3
N° discrete categories	29136
N° constraints	19

TABLE 5.4: Defining characteristics of the sub-problems comprising the multi-stage launch vehicle architecture optimization problem. (S: Solid propulsion, L: Liquid propulsion)

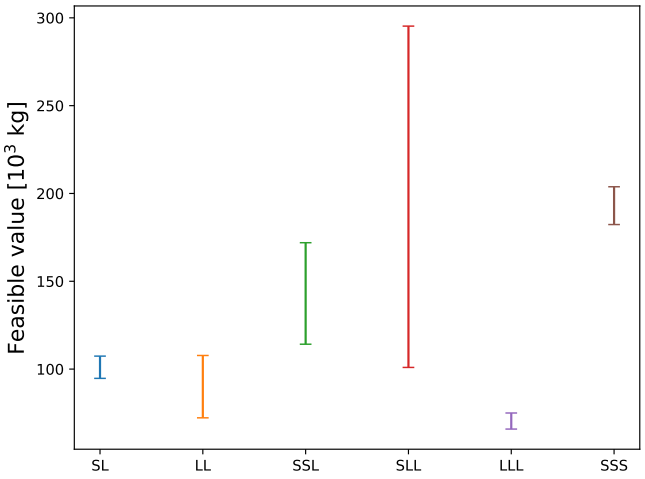


FIGURE 5.15: Value range of the feasible objective function for each sub-problem of the multi-stage launch vehicle architecture design problem.

5.4.3.4 Optimization results

The compared algorithms are initialized with a total data set of 122 data samples. Subsequently, the optimizations are performed by relying on 58 additional function evaluations. Please note that because of the complexity of the considered problem, the SOMVSP methods are only applied with values of a equal to 3, in order to reduce the chances of having optimal sub-problems discarded. The results obtained over 10 repetitions are provided in Figures 5.16 and 5.17.

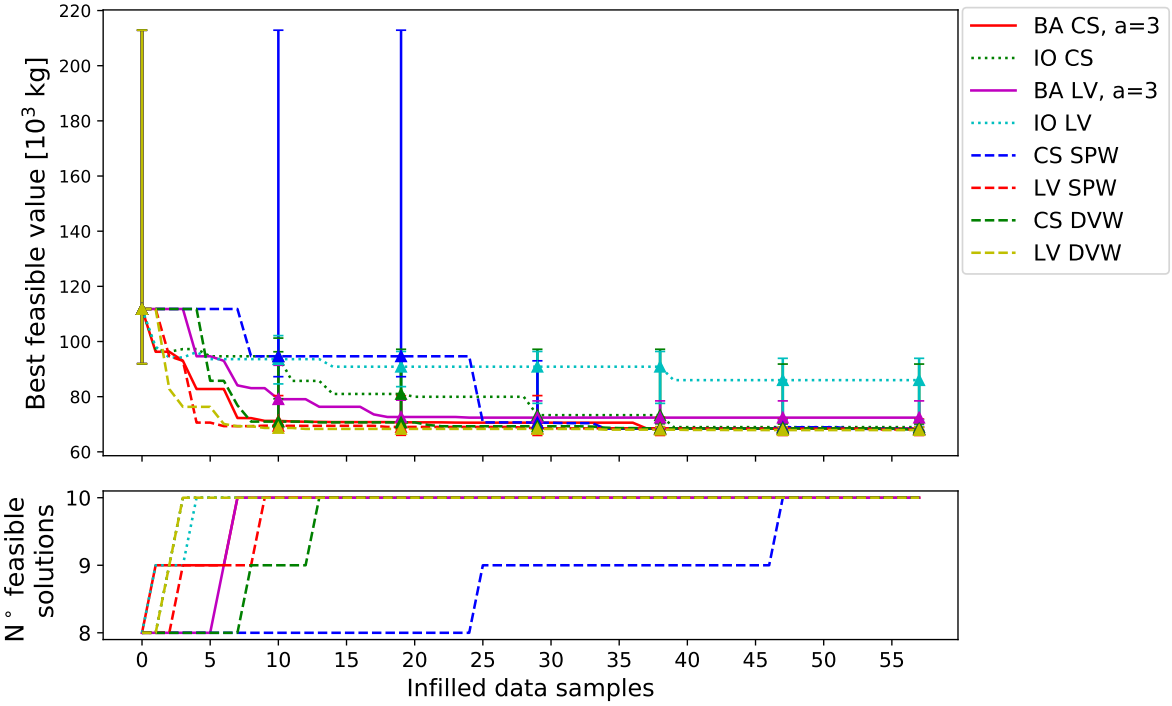


FIGURE 5.16: Comparison of the convergence rate of various discrete kernels during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. The bottom part of the plots represents the number of repetitions which have found a feasible solution at a given iteration.

A global analysis of the presented results shows that both the SOMVSP methods and the variable-size design space kernel based methods provide better optimization results when compared to the independent optimization of each sub-problem. This difference can be seen in terms of convergence speed as well as final optimal value obtained at the end of the optimization process. Indeed, when considering the results obtained at the end of the optimization process, it can be noticed that the independent optimization of each sub-problem is often not sufficient in order to identify the neighborhood of the global optimum, whereas for most of the repetitions, the proposed methods are able to do so. This can be explained by the fact that the proposed methods allow to better exploit the information provided by the initial data set in order to identify the most promising areas of the design space, as well as providing a more efficient and focused use of the available computational budget. In order to better compare the performance of the proposed methods, the convergence rate obtained with the SOMVSP and with the variable-size design space kernel BO are separately presented in Figures 5.18 and 5.19.

Similarly to what is obtained for the variable-size design space Goldstein test-case, Figure 5.19 shows a better performance of the Dimensional Variable-Wise approach when compared to the Sub-Problem-Wise approach in terms of convergence speed. This difference is due to the fact that the DVW kernel enables to exploit a larger amount of information when computing the covariance between samples which belong to different sub-problems. Furthermore, a slightly better performance of the LV kernel with respect to the CS one can be identified when considering the variable-size design space kernel approach, as it allows to model more complex trends by relying on a larger number of hyperparameters. An opposite trend can instead be identified when analyzing the results obtained with the SOMVSP method variants as well as for the independent optimization of each sub-problem. Indeed, for these approaches the CS kernel yields a faster and more consistent convergence if compared to the LV kernel. This can be explained by the fact that both approaches rely on the independent optimization of each sub-problem, which is performed with a considerably smaller amount of data. This makes the training of the LV hyperparameters

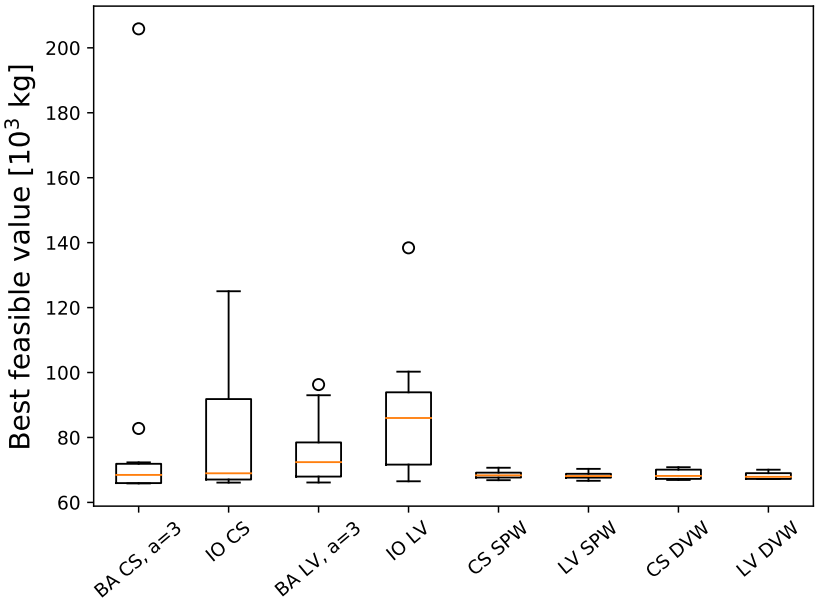


FIGURE 5.17: Comparison of the convergence value of different VSDSP optimization algorithms for the BO of the multi-stage launch vehicle architecture design over 10 repetitions.

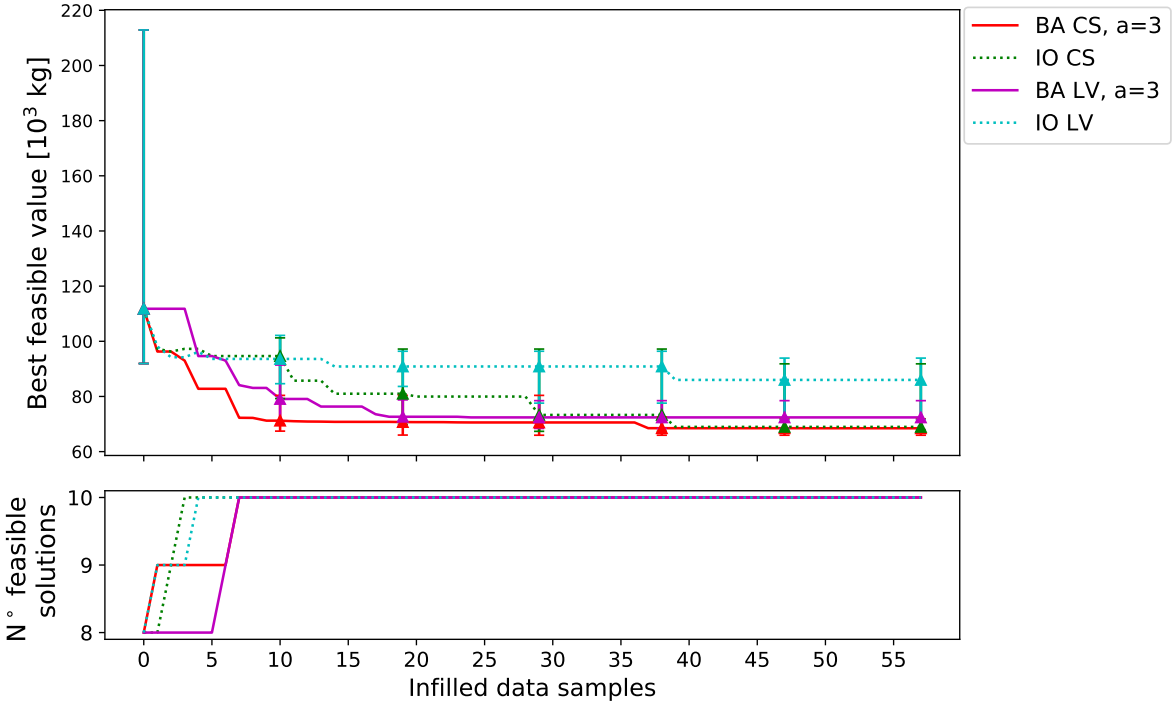


FIGURE 5.18: Comparison of the convergence rate of different VSDSP optimization algorithms during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. Focus on the SOMVSP methods. The bottom part of the plots represents the number of repetitions which have found a feasible solution at a given iteration.

more challenging, thus resulting in less accurate surrogate models.

A second noticeable difference between the two families of VSDSP optimization methods is related to the speed at which the the compared algorithms are able to identify the feasible areas of the search space. Indeed, it can be seen that for the repetitions which are not initialized with a

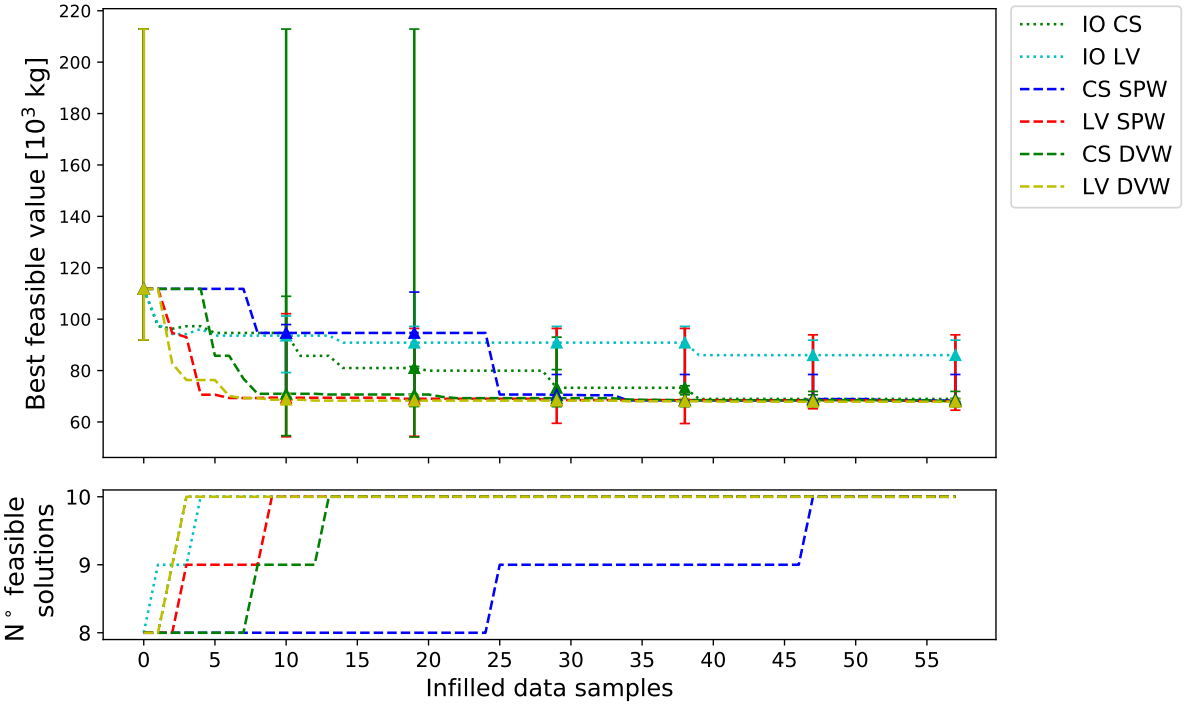


FIGURE 5.19: Comparison of the convergence rate of different VSDSP optimization algorithms during the BO of the multi-stage launch vehicle architecture design over 10 repetitions. Focus on the variable-size design space kernel methods. The bottom part of the plots represents the number of repetitions which have found a feasible solution at a given iteration.

feasible value within the data set, the SOMVSP methods are able to identify the feasible domain more rapidly if compared to the variable-size design space kernel based algorithms. This can be better explained by analyzing the number of remaining sub-problems along the SOMVSP process, as shown in Figure 5.20. It can be seen that for both the considered SOMVSP variants, all of the

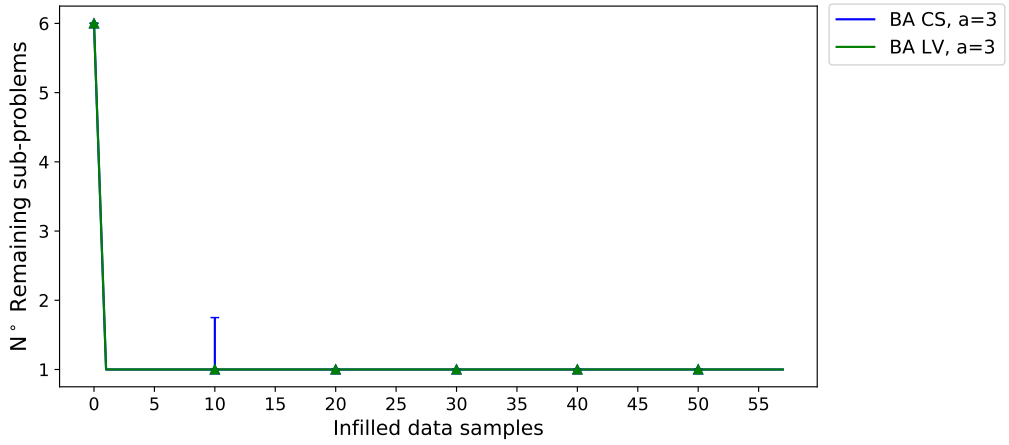


FIGURE 5.20: Comparison of the remaining number of sub-problems along the optimization process for different SOMVSP algorithms during the BO of the multi-stage launch vehicle architecture design over 10 repetitions.

sub-problems but one are discarded at the first iteration of nearly every repetition. Subsequently, having to deal with a single mixed-variable problem rather than with the global VSDSP allows

the SOMVSP to more easily identify the feasible areas of the design space, thus explaining the faster identification of the feasible areas of the design space.

The obtained results show that two families of VSDSP optimization algorithms yield similar convergence speeds, with a slight better performance for variable-size design space kernel based BO. However, Figure 5.17 shows that the SOMVSP methods provide a less robust convergence towards the actual optimum of the considered problem with respect to the initial data set. This can be explained with the help of Figure 5.21, in which the sub-problems towards which the different algorithms converge over the 10 repetitions are shown. It can be seen that the variable-size design space kernel based BO methods consistently convergence towards the triple liquid propulsion sub-problem, which contains the global optimum (see Figure 5.15). The SOMVSP methods, instead, also happen to converge towards non optimal sub-problems, such as the 2-stage solid/liquid propulsion architecture (SL) and the 2-stage liquid propulsion architecture (LL).

By combining the information provided by Figure 5.20 and Figure 5.21, it can be deduced that the SOMVSP method variants are not provided with sufficient data samples at the beginning of the optimization process. As a consequence, the sub-problems surrogate models are not accurate enough and the optimal sub-problems are discarded during the initial phases of the process, thus resulting in optimization runs which do not converge to the global optimum. A larger size initial data set could theoretically improve the robustness of the proposed method with and ensure a convergence towards the correct sub-problem. However, this would be incoherent with the scope of the thesis, which is related to the optimization of computationally intensive problems. Alternatively, the maximum accepted EV value (t) for the challenging constraints could manually be tuned in order to allow for a larger violation tolerance at the beginning of the optimization process, and subsequently be reduced once sufficient data is provided. However, this would require prior information on the considered problem and time-consuming tuning in order to ensure a consistent and fast convergence of the optimization process.

5.4.4 Result synthesis

The results obtained for the optimization of the variable-size design space Goldstein function and the multi-stage launch vehicle architecture design show that the two alternative methods proposed in order to solve VSDSP provide overall a considerably faster and more consistent convergence towards the problem optimum when compared to the independent optimization of each sub-problem. This difference is due to the fact that the proposed methods allow to exploit more efficiently the information provided by the initial data set, and are therefore able to focus the available computational budget towards the areas of interest of the design space. Additionally, the results show a slightly better performance of the variable-size design space kernel approach when compared to the SOMVSP one, both in terms of convergence speed and robustness with respect to the initial data set. Again, this can be explained by the fact that the variable-size design space kernel based BO can exploit the entirety of the data set in order to identify the most promising areas of the design space, whereas the GP models used by the SOMVSP are independently defined over the various sub-problems and rely therefore on smaller data sets. However, the results also show that if the considered SOMVSP can be initialized with a suitable data set, in terms of size and available information, it is able to efficiently identify and discard the majority of non-optimal sub-problems. In this case, the SOMVSP is only required to solve a limited number of lower dimension mixed-variable problems rather than the global VSDSP, and can therefore easily identify the areas of interest of the variable-size design space without actually having to explore it all.

The proposed variable-size design space kernel based methods (*i.e.,* SPW and VDW) present a fast convergence speed as well as a good robustness with respect to the initial data set for both considered test-cases. Furthermore, the obtained results also show that between the two variants, the dimensional variable-wise decomposition yields a faster convergence speed due to the fact that

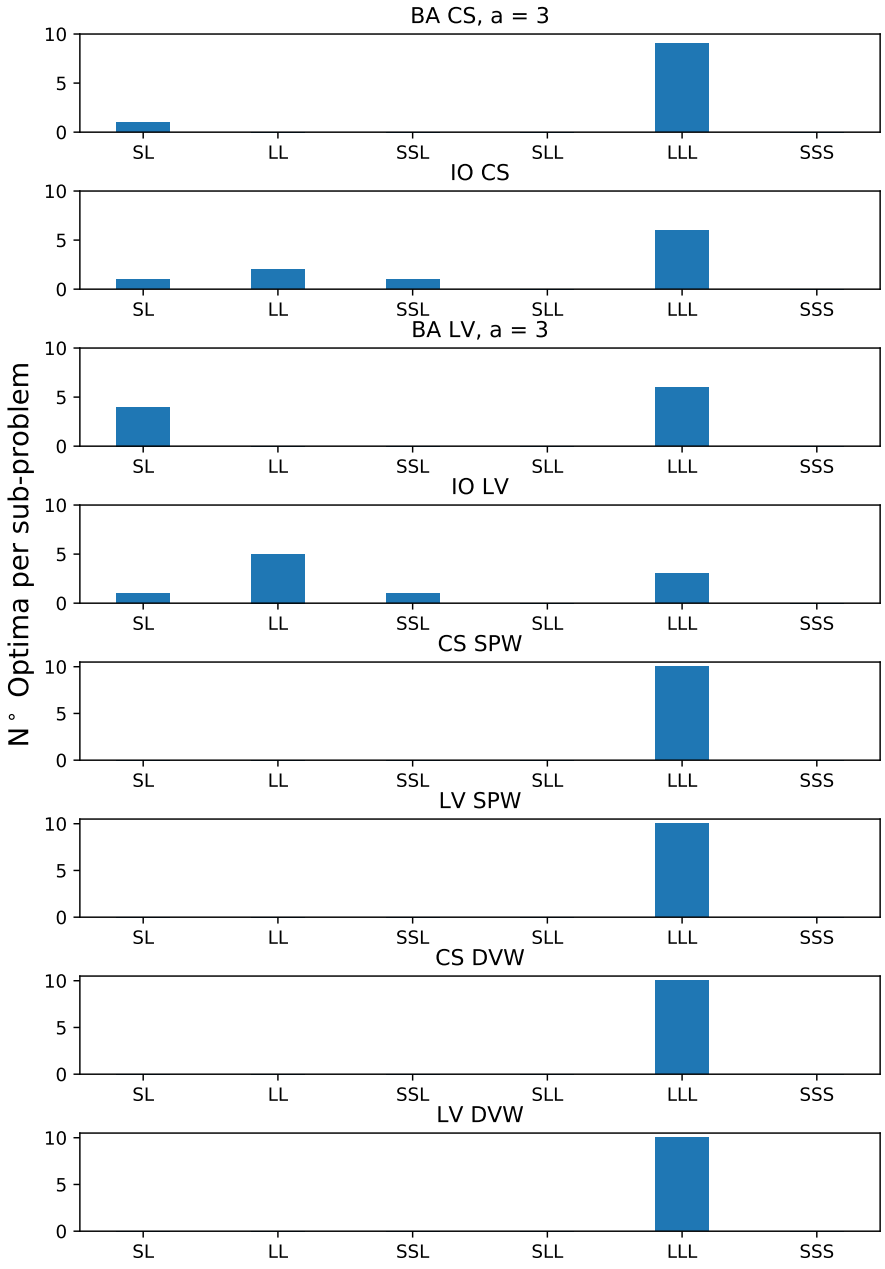


FIGURE 5.21: Sub-problems towards which the compared VSDSP algorithms convergence over the 10 repetitions of the multi-stage launch vehicle architecture design optimization.

a larger amount of information can be exploited when computing the covariance between samples which belong to different sub-problems. The sub-problem-wise decomposition approach, instead, only relies on a kernel defined with respect to the (scalar) dimensional variable levels in order to compute the same type of covariance value. A second noticeable trend when analyzing the results obtained with variable-size design space kernel based BO methods is that, when confronted with problems characterized by large discrete combinatorial spaces, the LV kernel tends to yield a faster convergence towards the problem optimum when compared to the CS one, as is for instance shown for the multi-stage launch vehicle architecture design test-case. This difference in performance is related to the limited modeling capabilities of the CS kernel when modeling discrete or dimensional variables characterized by a large number of levels. For example, the CS kernel based SPW approach relies on a single covariance value between any pair of different

sub-problems, which tends to be overly simplistic when dealing with complex variable-size design space functions.

The proposed SOMVSP method shows a promising convergence speed as well, if compared to the independent optimization of each sub-problem. However, the obtained results also indicate that this approach is less robust with respect to the initial provided data set, which results in a larger variance of the determined optimum values. This can be explained by the fact that the SOMVSP relies on independent GP models for each sub-problem, which are created by relying on small-sized data sets. As a consequence, the surrogate model that are used in order to determine how promising each sub-problem is might be inaccurate, thus resulting in the premature discarding of the optimal sub-problem. This issue can be particularly problematic when confronted with problems characterized by a large number of constraints, such as the multi-stage launch vehicle architecture design. Indeed, in these cases the feasible domain of the design space can become challenging to model and identify, which subsequently leads the SOMVSP algorithm to make the wrong choices with respect to the budget allocation and sub-problem discarding. This issue could be partially avoided by providing larger data sets at the beginning of the process and/or by having the EV threshold vary along the optimization as a function of the constraint models accuracy. However, these solutions would be computationally expensive and/or would require problem specific and time consuming tuning. Finally, the results obtained for the variable-size design space Goldstein function show that lower values of the SOMVSP parameter a result in a faster discarding of sub-problems, which in turn yields a faster convergence. However, low values of a can also cause the premature discarding of optimal sub-problems during at first phases of the optimization process, during which the surrogate models are created with a limited amount of data.

5.5 Conclusions

In this chapter, two alternative extensions of the mixed-variable Bayesian optimization method presented in Chapter 4 are proposed in order to perform the optimization of variable-size design space problems. The first approach, referred to as SOMVSP, is a budget allocation strategy based on the independent BO of each sub-problem characterizing the considered VSDSP. The second approach, instead, is based on the definition of a variable-size design space kernel allowing to compute the covariance between samples characterized by partially different sets of variables. The two proposed VSDSP optimization methods are tested on an analytical test-case as well as on an engineering related problem, namely the architecture design of a multi-stage launch vehicle. Overall, the obtained results show that the proposed algorithms provide a better optimization performance in terms of convergence speed as well as robustness with respect to the initial data set when compared to the independent optimization of each sub-problem, which is the standard approach when dealing with variable-size design space problems within the framework of complex system design. Furthermore, it is shown that for the considered test-cases, the variable-size design space kernel BO approach performs better when compared to the SOMVP due to the fact that the considered GP models are constructed by relying on the entirety of the data set, thus providing a more accurate modeling of the objective and constraint functions. The SOMVSP also provides a good convergence speed if compared to the independent optimization of each sub-problem. However, it presents a larger variance in terms of the results obtained at the end of the optimization process. Indeed, the obtained results show that the global optimum is not identified at every optimization repetition of the SOMVSP variants. This phenomenon is explained by the fact that when insufficient data is provided to the optimization algorithm, inaccurate modeling can result in the discarding of the sub-problem containing the global optimum. Additionally, the discarding of the sub-problems is performed by considering different scenarios of the feasible predicted optimum, which is determined by relying on the nominal prediction of the constraint

function values. This can further increase the likelihood of optimal sub-problems being discarded in cases in which the constraints are inaccurately modeled or when dealing with a large number of constraints. As a consequence, it could be valuable to develop alternative definitions of the best, worst and nominal case scenarios which include a margin of safety with respect to the predicted constraint function values in order to improve the robustness of the SOMVSP with respect to the initial data set.

The optimization algorithms proposed in this chapter provide a solution for the optimization of variable-size design space problems by requiring a relatively limited number of function evaluations. As a consequence, they allow dealing with complex system design problems which depend on technological choices and which are characterized by computationally intensive simulation codes in their most general formulation (*i.e.,* without simplifications and/or assumptions). Among the two alternative proposed approaches, the variable-size design space kernel based BO is more suitable when dealing with purely black-box problems for which no prior information is known, as it requires no user-defined parameterization and is more robust to the initial data set. The SOMVSP, instead, can be efficiently used in order to identify a limited number of promising system architectures, which can then be independently analyzed, and if necessary optimized.

CHAPTER 6

Conclusions and perspectives

6.1 Conclusions

In this thesis, the possibility of including the presence of discrete technological choices within a Bayesian optimization framework is discussed, thus allowing to perform the modeling and optimization of constrained mixed-variable problems and constrained variable-size design space problems. The driving purpose of the presented research is enabling the identification and definition of optimal architectures within the early stages of the design of a complex system while requiring a limited number of computationally intensive numerical simulations. Throughout this work, the design of launch vehicles is considered as an illustrative example in order to better highlight the advantages and limitations of the proposed algorithms and methods. However, it is important to note that they are applicable to a much wider class of complex system design problems. The most notable results and conclusions which can be drawn from the presented work are discussed in the following paragraphs.

First, it is shown that a complex system design problem characterized by the presence of technological choices can be generalized under the form of a so-called variable-size design space problem. These particular problems present a simultaneous dependence on 3 different types of design variables: continuous, discrete and dimensional. Dimensional variables are similar to discrete variables, with the main distinction being that depending on their value, the number and type of continuous and discrete variables the problem functions depend on can vary, as well as the number and type of constraints the problem is subject to. The presented review of the literature discussing the algorithms allowing to solve this particular type of problem shows that none of the existing methods provides a reliable and efficient solution when confronted to computationally intensive problems due to the large number of required function evaluations as well as the inadequate handling of constraints. As a consequence, the necessity of relying on Bayesian optimization in order to avoid these limitations is identified.

In order to analyze the possibility of performing the Bayesian optimization of variable-size design space problems, the Gaussian process based surrogate modeling of (fixed-size) functions depending on continuous and discrete variables is first discussed. It is shown that a kernel in the mixed-variable design space can be defined as a product between purely continuous and purely discrete kernels. A unifying formalism allowing to construct valid kernels in the discrete design space is presented, and is subsequently used in order to compare the existing discrete kernels from a theoretical perspective for modeling purposes. The actual modeling performance of these kernels is then tested on several benchmark functions with different characteristics. Overall, the

obtained results show that performing the surrogate modeling of mixed-variable functions by relying on a single mixed-variable kernel rather than on multiple independent continuous ones allows to provide a considerably more accurate modeling. Furthermore, the results indicate that no universally better discrete kernel parameterization exists, as their relative modeling performance varies depending on the considered mixed-variable function characteristics, such as the discrete design space combinatorial size, the function scedasticity and the presence of linear and/or negative correlation trends. Furthermore, the obtained kernels relative performance is also heavily influenced by the amount of data provided for the creation of the surrogate model as a function of the number of hyperparameters to be trained. Finally, it is worth noticing that because of the way they are constructed, part of the discussed discrete kernels results in a more challenging model training when compared to purely continuous Gaussian processes due to the necessity of a good initialization of the hyperparameter values. In these cases, a multiple random initialization of the acquisition function is required, thus considerably increasing the model training computational overhead.

Having analyzed the Gaussian process based surrogate modeling of mixed-variable functions, the possibility of relying on such models in order to perform the Bayesian optimization of (fixed-sized) constrained mixed continuous/discrete problems is subsequently discussed. It is shown that acquisition functions commonly used for purely continuous problems, such as the expected improvement, are still valid in the mixed-variable case under the condition of considering a valid kernel, with the only additional requirement of performing the acquisition function optimization in the mixed-variable design space. The results obtained by testing the proposed algorithm on several benchmark problems show the benefits of relying on a single mixed-variable kernel rather than on multiple independent continuous ones in terms of both convergence speed and optimal solution value. Furthermore, a considerably better optimization performance with respect to standard heuristic algorithms is also shown. Additionally, the results indicate that the dependence of the optimization performance on the considered discrete kernel parameterization is less important when compared to the previously discussed modeling context. Indeed, simpler and linear kernels tend to provide similar convergence speeds when compared to more complex ones thanks to a larger robustness to local over-fitting phenomena. For the same reasons, in some particular engineering related test-cases characterized by linear trends and low amounts of available data, these linear kernels provide better results than more complex and non-linear kernels. Finally, it is worth mentioning that the majority of the objective and constraint functions considered in order to test the optimization performance of the proposed algorithm present relatively smooth trends. On the other hand, due to the very nature of Gaussian process surrogate modeling, the presented method might yield slower and/or less consistent results when confronted with non stationary objective and constraints as a result of a less accurate modeling of these functions.

In the last part of the thesis, the possibility of extending the proposed mixed-variable Bayesian optimization algorithm in order to solve variable-size design space problems is discussed. More specifically, two alternative approaches are proposed. The first one is based on the simultaneous optimization of several fixed-size mixed-variable sub-problems coupled with a budget allocation strategy, allowing to focus the computational effort onto the most promising sub-problems and discard the least promising ones. The second proposed approach is instead based on the definition of a Gaussian process kernel allowing to compute the covariance between samples characterized by partially different sets of variables by performing a hierarchical grouping of the design variables. By defining this kernel, it is then possible to perform the direct Bayesian optimization of variable-size design space problems, with the additional challenge of dealing with acquisition functions defined within the same variable-size design space. The benchmarking of the two proposed optimization algorithms on two different test-cases shows that both methods allow to provide a considerably faster convergence to the optimum when compared to the independent

optimization of each sub-problem. Depending on the characteristics of the considered optimization problem, and more specifically on the difference in terms of feasibility domain and objective function value between the various sub-problems, the relative performance of the two proposed approaches varies. In general, the results indicate that the budget allocation strategy provides a useful tool for comparing and selecting the most promising system architectures when a clear difference in performance between the various sub-problems exists. Instead, the variable-size design space kernel based Bayesian optimization represents a more robust approach when dealing with sub-problems with similar performances and/or inaccurately modeled constraints.

Overall, the Bayesian optimization algorithms proposed for both fixed-size and variable-size design space problems provide a considerably faster convergence towards the considered problem optimum (in terms of number of functions evaluations) when compared to standard engineering approaches as well as reference heuristic algorithms. They represent therefore a promising tool for the design optimization of complex systems in the presence discrete technological choices and computationally intensive objective and/or constraint functions as they would allow to reduce the amount of time required in order to define a baseline architecture. Alternatively, they could also enable the exploration of a larger number of possible system architectures and configurations. However, it is also important to note that Gaussian process surrogate modeling suffers from the curse of dimension, and tends to be inadequate when confronted with optimization problems characterized by a a large number of design variables and/or large amount of training data.

6.2 Perspectives

Different possible improvements and/or extensions of the work presented in this thesis can be identified. First of all, the proposed mixed-variable and variable-size design space Bayesian optimization algorithms are only defined within the context of single-objective optimization problems. However, real-life design optimization problems are usually characterized by the presence of multiple antagonistic objectives, such as the simultaneous requirement of low production costs and high performance. For this reason, it could be valuable to assess the possibility of adapting and extending existing multi-objective acquisition functions, such as the expected hypervolume improvement, in order to enable the optimization of multi-objective mixed-variable problems and variable-size design space problems.

The proposed formulation of the variable-size design space kernel allowing to compute the covariance between data samples characterized by partially different sets of variables is based on the assumption that the dimensional variable design space does not vary as a function of the dimensional variables themselves. Indeed, based on this assumption, the hierarchical grouping of continuous and discrete design variables with respect to the dimensional ones can be performed. However, in some particular cases this assumption may not be true, which would then require to handle the presence of nested dimensional variables. For this reason, a generalization of the variable-size design space kernel based Bayesian optimization algorithm allowing to solve this particular type of problems could allow to further widen its possible applications and relax the constraints on the problem formulation.

In order to reduce their global computational cost, real-life complex system design problems often require to rely on different sets of samples that characterize the same measurable quantity but which are computed with different fidelities (*i.e.,* different levels of approximation). Additionally, these different data sets can sometimes be characterized by partially different continuous and discrete design variables. Typically, higher fidelity computations tend to depend on a larger number of design variables. By its very nature, the proposed variable-size design space kernel provides a

solution for the modeling of such multi-fidelity problems. It could therefore be interesting to analyze the possibility of performing mixed-variable multi-fidelity modeling by relying on Gaussian processes (*e.g.,* co-Kriging) coupled with the proposed mixed-variable and variable-size design space kernels.

Real-life complex system design problems are usually characterized by several interacting disciplines which tend to have antagonistic effects on the system performance and can therefore result in conflicting decisions and/or unfeasible solutions. For this reason, multidisciplinary optimization problems are normally solved with the help of Multidisciplinary Design Optimization (MDO) formulations, which provide efficient ways of handling the couplings between disciplines and therefore facilitate the research of the optimal compromise. Like other approaches, MDO formulations tend to provide poor results when confronted with computationally intensive problems due to the large number of function evaluations required in order to determine feasible compromises between disciplines. For this reason, it could be valuable to study the possibility of coupling the Bayesian optimization algorithms presented in this thesis with existing MDO formulations in order to reduce the computational effort required to solve MDO problems.

Finally, it can be noticed that the methods and algorithms proposed in this thesis all provide a solution for global optimization problems without any requirement of prior knowledge for the initialization of the optimization process. As a consequence, they allow to efficiently determine the discrete and dimensional categories which characterize the optimal solution as well as the location of the global optimum continuous neighborhood in said categories. For these reasons, the performance of the proposed algorithms when dealing with real design optimization problems could be further improved if they were to be coupled with local continuous optimization methods, such as gradient-based algorithms, thus allowing to refine the incumbent solution in the continuous design space and provide a convergence towards the actual global optimum of the considered problem.

Bibliography

[1] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[2] O. Abdelkhalik. "Autonomous planning of multigravity-assist trajectories with deep space maneuvers using a differential evolution approach". In: *International Journal of Aerospace Engineering* 2013 (2013), pp. 1–11. ISSN: 16875966. DOI: 10.1155/2013/145369.

[3] O. Abdelkhalik. "Hidden Genes Genetic Optimization for Variable-Size Design Space Problems". In: *Journal of Optimization Theory and Applications* 156.2 (2013), pp. 450–468. ISSN: 00223239. DOI: 10.1007/s10957-012-0122-6.

[4] M. A. Abramson, C. Audet, J. W. Chrissis, and J. G. Walston. "Mesh adaptive direct search algorithms for mixed variable optimization". In: *Optimization Letters* 3.1 (2009), pp. 35–47. ISSN: 18624472. DOI: 10.1007/s11590-008-0089-2.

[5] M. A. Abramson, C. Audet, and J. E. Dennis Jr. "Filter pattern search algorithms for mixed variable constrained optimization problems". In: *Pacific Journal of Optimization* 3.3 (2007), pp. 477–500.

[6] M. A. Alvarez, L. Rosasco, N. D. Lawrence, et al. "Kernels for vector-valued functions: A review". In: *Foundations and Trends® in Machine Learning* 4.3 (2012), pp. 195–266.

[7] N. Aronszajn. "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

[8] C. Audet, J Denni, D. Moore, A. Booker, and P. Frank. "A surrogate-model-based method for constrained optimization". In: *8th Symposium on Multidisciplinary Analysis and Optimization*. 2000, p. 4891.

[9] C. Audet and J. E. Dennis Jr. "Pattern Search Algorithms For Mixed Variable Programming". In: *SIAM Journal on Optimisation* 11.3 (2000), pp. 573–594. ISSN: 1052-6234. DOI: 10.1137/S1052623499352024.

[10] L. Bajer and M. Holeňa. "Surrogate model for mixed-variables evolutionary optimization based on GLM and RBF networks". In: *Lecture Notes in Computer Science*. Vol. 7741 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 481–490. ISBN: 9783642358425. DOI: 10.1007/978-3-642-35843-2_41.

[11] L. Bajer and M. Holeňa. "Surrogate model for continuous and discrete genetic optimization based on RBF networks". In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer. 2010, pp. 251–258.

[12] M. Balesdent. "Multidisciplinary Design Optimization of Launch Vehicles". PhD thesis. Ecole Centrale de Nantes (ECN), 2011.

[13] M. Balesdent, N. Bérend, P. Dépincé, and A. Chriette. *A survey of multidisciplinary design optimization methods in launch vehicle design*. 2012. DOI: 10.1007/s00158-011-0701-4.

[14] M. Balesdent, L. Brevault, N. B. Price, S. Defoort, R. Le Riche, N.-H. Kim, R. T. Haftka, and N. Bérend. "Advanced Space Vehicle Design Taking into Account Multidisciplinary Couplings and Mixed Epistemic/Aleatory Uncertainties". In: Springer, Cham, 2016, pp. 1–48. DOI: 10.1007/978-3-319-41508-6_1.

[15] P.-J. Barjhoux, Y. Diouane, S. Grihon, D. Bettebghor, and J. Morlier. "Mixed variable Structural optimization: toward an efficient hybrid algorithm". In: *World Congress of Structural and Multidisciplinary Optimisation*. Springer. 2017, pp. 1880–1896.

[16] T. Bartz-Beielstein and M. Zaefferer. "Model-based methods for continuous and discrete global optimization". In: *Applied Soft Computing* 55 (2017), pp. 154–167. ISSN: 15684946. DOI: 10.1016/j.asoc.2017.01.039.

[17] C. Beauthier, A. Mahajan, C. Sainvitu, P. Hendrick, S. Sharifzadeh, and D. Verstraete. "Hypersonic cryogenic tank design using mixed-variable surrogate-based optimization". In: *Engineering Optimization IV - Proceedings of the 4th International Conference on Engineering Optimization, ENGOPT 2014*. CRC Press, 2014, pp. 543–549. ISBN: 9781138027251. DOI: 10.1201/b17488-98.

[18] H.-G. Beyer and H.-P. Schwefel. "Evolution strategies–A comprehensive introduction". In: *Natural computing* 1.1 (2002), pp. 3–52.

[19] B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. "mlr: Machine Learning in R". In: *Journal of Machine Learning Research* 17.170 (2016), pp. 1–5.

[20] J.-F. Bonnans, J. C. Gilbert, and C. Lemarechal. *Numerical optimization: theoretical and practical aspects*. Springer Berlin Heidelberg, 2006, p. 423. ISBN: 3662050781.

[21] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier. "Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction". In: *Structural and Multidisciplinary Optimization* 53.5 (2016), pp. 935–952.

[22] P. Breitkopf and R. F. Coelho. *Multidisciplinary design optimization in computational mechanics*. John Wiley & Sons, 2013.

[23] R. K. Brouwer. "A feed-forward network for input that is both categorical and quantitative". In: *Neural Networks* 15.7 (2002), pp. 881–890.

[24] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[25] A. Christmann and R. Hable. "Consistency of support vector machines using additive kernels for additive models". In: *Computational Statistics & Data Analysis* 56.4 (2012), pp. 854–873.

[26] P. Craven and G. Wahba. "Smoothing noisy data with spline functions". In: *Numerische mathematik* 31.4 (1978), pp. 377–403.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. ISSN: 1089778X. DOI: 10.1109/4235.996017.

[28] X. Deng, Y. Hung, and C. D. Lin. "Design for computer experiments with qualitative and quantitative factors". In: *Statistica Sinica* (2015), pp. 1567–1581.

[29] X. Deng, C. D. Lin, K. W. Liu, and R. K. Rowe. "Additive Gaussian Process for Computer Models With Qualitative and Quantitative Factors". In: *Technometrics* 59.3 (2017), pp. 283–292. ISSN: 15372723. DOI: 10.1080/00401706.2016.1211554.

[30] C. R. Dietrich and M. R. Osborne. "Estimation of covariance parameters in kriging via restricted maximum likelihood". In: *Mathematical Geology* 23.1 (1991), pp. 119–135. ISSN: 0882-8121. DOI: 10.1007/BF02065971.

[31] M. Dorigo. "Optimization, learning and natural algorithms". PhD thesis. Politecnico di Milano, Italy, 1992. DOI: citeulike-article-id:6415913.

[32] N. R. Draper and H. Smith. *Applied regression analysis*. Vol. 326. John Wiley & Sons, 1998.

[33] C. Durantin, J. Marzat, and M. Balesdent. "Analysis of multi-objective Kriging-based methods for constrained global optimization". In: *Computational Optimization and Applications* 63.3 (2016), pp. 903–926. ISSN: 0926-6003. DOI: 10.1007/s10589-015-9789-6.

[34] N. Dyn, D. Levin, and S. Rippa. "Numerical procedures for surface fitting of scattered data by radial functions". In: *SIAM Journal on Scientific and Statistical Computing* 7.2 (1986), pp. 639–659.

[35] M. Emmerich, M. Grötzner, B. Groß, and M. Schütz. "Mixed-Integer Evolution Strategy for Chemical Plant Optimization with Simulators". In: *Evolutionary Design and Manufacture*. London: Springer London, 2000, pp. 55–67. DOI: 10.1007/978-1-4471-0519-0_5.

[36] H. Fang and M. F. Horstemeyer. "Global response approximation with radial basis functions". In: *Engineering Optimization* 38.04 (2006), pp. 407–424.

[37] R. Filomeno Coelho. "Metamodels for mixed variables based on moving least squares". In: *Optimization and Engineering* 15.2 (2014), pp. 311–329. ISSN: 1389-4420. DOI: 10.1007/s11081-013-9216-8.

[38] E. Fitkov-Norris, S. Vahid, and C. Hand. "Evaluating the impact of categorical data encoding and scaling on neural network classification performance: the case of repeat consumption of identical cultural goods". In: *International Conference on Engineering Applications of Neural Networks*. Springer. 2012, pp. 343–352.

[39] R. Fletcher. "Practical methods of optimization". In: (2013).

[40] H. Ford and J. M. Alexander. *Advanced mechanics of materials*. E. Horwood, 1977, p. 672. ISBN: 0470990651.

[41] A. Forrester and A. Keane. "Recent advances in surrogate-based optimization". In: *Progress in aerospace sciences* 45.1-3 (2009), pp. 50–79.

[42] A. Forrester, A. Sobester, and A. Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[43] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. "DEAP: Evolutionary Algorithms Made Easy". In: *Journal of Machine Learning Research* 13 (2012), pp. 2171–2175.

[44] C. Frank, R. Marlier, O. J. Pinon-Fischer, and D. N. Mavris. "An Evolutionary Multi-Architecture Multi-Objective Optimization Algorithm for Design Space Exploration". In: *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, 2016. ISBN: 978-1-62410-392-6. DOI: 10.2514/6.2016-0414.

[45] C. P. Frank. "A design space exploration methodology to support decisions under evolving uncertainty in requirements and its application to advanced vehicles". In: (2016).

[46] J. H. Friedman et al. "Multivariate adaptive regression splines". In: *The annals of statistics* 19.1 (1991), pp. 1–67.

[47] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications*. Vol. 20. Siam, 2007.

[48] D. E. Goldberg and Addison-Wesley. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989, p. 412. ISBN: 0201157675.

[49] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[50] P. Goovaerts et al. *Geostatistics for natural resources evaluation*. Oxford University Press on Demand, 1997.

[51] J. C. Gower. "A General Coefficient of Similarity and Some of Its Properties". In: *Biometrics* 27.4 (1971), p. 857. ISSN: 0006341X. DOI: 10.2307/2528823.

[52] R. B. Gramacy and H. K. H. Lee. "Bayesian Treed Gaussian Process Models With an Application to Computer Modeling". In: *Journal of the American Statistical Association* 103.483 (2008), pp. 1119–1130. ISSN: 0162-1459. DOI: 10.1198/016214508000000689.

[53] R. T. Haftka, E. P. Scott, and J. R. Cruz. "Optimization and Experiments: A Survey". In: *Applied Mechanics Reviews* 51.7 (1998), pp. 435–448. ISSN: 00036900. DOI: 10.1115/1.3099014.

[54] M. Halstrup. "Black-box optimization of mixed discrete-continuous optimization problems". PhD thesis. TU Dortmund, 2016. DOI: 10.17877/DE290R-17800.

[55] N. Hansen. "Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms". In: *Towards a New Evolutionary Computation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. ISBN: 978-3-540-32494-2. DOI: 10.1007/3-540-32494-1_4.

[56] R. L. Hardy. "Multiquadric equations of topography and other irregular surfaces". In: *Journal of geophysical research* 76.8 (1971), pp. 1905–1915.

[57] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[58] H. C. Herbol, W. Hu, P. Frazier, P. Clancy, and M. Poloczek. "Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization". In: *npj Computational Materials* 4.1 (2018), p. 51.

[59] M. Herrera, A. Guglielmetti, M. Xiao, and R. Filomeno Coelho. "Metamodel-assisted optimization based on multiple kernel regression for mixed variables". In: *Structural and Multidisciplinary Optimization* 49.6 (2014), pp. 979–991. ISSN: 1615-147X. DOI: 10.1007/s00158-013-1029-z.

[60] K. Holmström, N. H. Quttineh, and M. M. Edvall. "An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization". In: *Optimization and Engineering*. Vol. 9. Springer US, 2008, pp. 311–339. ISBN: 1389-4420. DOI: 10.1007/s11081-008-9037-3.

[61] C.-C. Hsu. "Generalizing self-organizing map for categorical data". In: *IEEE transactions on Neural Networks* 17.2 (2006), pp. 294–304.

[62] F. Hutter. "Automated configuration of algorithms for solving hard computational problems". PhD thesis. University of British Columbia, 2009.

[63] D. R. Jones, M. Schonlau, and W. J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13 (1998), pp. 455–492.

[64] A. G. Journel and C. J. Huijbregts. *Mining geostatistics*. Vol. 600. Academic press London, 1978.

[65] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995, pp. 1942–1948. ISBN: 0-7803-2768-3. DOI: 10.1109/ICNN.1995.488968.

[66] M. Kokkolaras, C. Audet, and J. E. Dennis Jr. "Mixed Variable Optimization of the Number and Composition of Heat Intercepts in a Thermal Insulation System". In: *Optimization and Engineering* 2.1 (2001), pp. 5–29. ISSN: 1389-4420. DOI: 10.1023/A:1011860702585.

[67] J. Lampinen and I. Zelinka. "Mixed integer-discrete-continuous optimization by differential evolution". In: *Proceedings of the 5th International Conference on Soft Computing*. 1999, pp. 71–76.

[68] P. Lancaster and K. Salkauskas. "Surfaces generated by moving least squares methods". In: *Mathematics of computation* 37.155 (1981), pp. 141–158.

[69] A. H. Land and A. G. Doig. "An Automatic Method for Solving Discrete Programming Problems". In: *50 Years of Integer Programming 1958-2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 105–132. DOI: 10.1007/978-3-540-68279-0_5.

[70] W. J. Larson, G. N. Henry, and R. W. Humble. *Space propulsion analysis and design*. McGraw-Hill, 1995.

[71] I. N. Levine. *Physical chemistry*. McGraw-Hill, 2009, p. 989. ISBN: 9780072538625.

[72] Q. Li and J. S. Racine. *Nonparametric econometrics: theory and practice*. Princeton University Press, 2007.

[73] R. Li, M. T. M. Emmerich, J. Eggermont, E. G. Bovenkamp, T. Bäck, J. Dijkstra, and J. H. Reiber. "Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis". In: *2008 IEEE Congress on Evolutionary Computation, CEC 2008*. Hong Kong: IEEE, 2008, pp. 2764–2771. ISBN: 9781424418237. DOI: 10.1109/CEC.2008.4631169.

[74] C.-J. Liao, Chao-Tang Tseng, and P. Luarn. "A discrete version of particle swarm optimization for flowshop scheduling problems". In: *Computers & Operations Research* 34.10 (2007), pp. 3099–3111. ISSN: 03050548. DOI: 10.1016/j.cor.2005.11.017.

[75] L. Liao and W. S. Noble. "Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships". In: *Journal of computational biology* 10.6 (2003), pp. 857–868.

[76] T. Liao, K. Socha, M. A. Montes de Oca, T. Stutzle, and M. Dorigo. "Ant Colony Optimization for Mixed-Variable Optimization Problems". In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 503–518. ISSN: 1089-778X. DOI: 10.1109/TEVC.2013.2281531.

[77] M. Liu, M. Dong, and C. Wu. "A new ANFIS for parameter prediction with numeric and categorical inputs". In: *IEEE Transactions on Automation Science and Engineering* 7.3 (2010), pp. 645–653.

[78] D. L. Logan. *A first course in the finite element method*. Cengage Learning, 2011.

[79] S. Lucidi, V. Piccialli, and M. Sciandrone. "An Algorithm Model for Mixed Variable Programming". In: *SIAM Journal on Optimization* 15.4 (2005), pp. 1057–1084. ISSN: 1052-6234. DOI: 10.1137/S1052623403429573.

[80] S. Ma, J. S. Racine, and L. Yang. "Spline regression in the presence of categorical predictors". In: *Journal of Applied Econometrics* 30.5 (2015), pp. 705–717.

[81] D. J. MacKay and D. J. Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[82] R. H. MacNeal and C. W. McCormick. "The NASTRAN Computer Program for Structural Analysis". In: *SAE Technical Paper*. SAE International, 1969. DOI: 10.4271/690612.

[83] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. "GPflow: A Gaussian process library using TensorFlow". In: *Journal of Machine Learning Research* 18.40 (2017), pp. 1–6.

[84] B. J. McBride and S. Gordon. *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications. User Manual and Program Description*. Tech. rep. NASA, 1996.

[85] P. McCullagh. *Generalized linear models*. Routledge, 2019.

[86] M. D. McKay, R. Beckman, and W. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (1979), p. 239. ISSN: 00401706. DOI: 10.2307/1268522.

[87] M. Meckesheimer, R. R. Barton, T. Simpson, F. Limayem, and B. Yannou. "Metamodeling of Combined Discrete/Continuous Responses". In: *AIAA JOURNAL* 39.10 (2001). DOI: 10.2514/2.1185.

[88] B. Minasny and A. B. McBratney. "The Matérn function as a general model for soil variograms". In: *Geoderma*. Vol. 128. Elsevier, 2005, pp. 192–207. ISBN: 00167061. DOI: 10.1016/j.geoderma.2005.04.003.

[89] J. Mockus. "Application of Bayesian approach to numerical methods of global and stochastic optimization". In: *Journal of Global Optimization* 4.4 (1994), pp. 347–365.

[90] A. Moraglio and A. Kattan. "Geometric generalisation of surrogate model based optimisation to combinatorial spaces". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer. 2011, pp. 142–154.

[91] J. Müller, C. A. Shoemaker, and R. Piché. "SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems". In: *Computers and Operations Research* 40.5 (2013), pp. 1383–1400. ISSN: 03050548. DOI: 10.1016/j.cor.2012.08.022.

[92] J. A. Nelder and R. Mead. "A simplex method for function minimization". In: *Computer Journal* 7.4 (1964), pp. 308–313. ISSN: 00104620. DOI: 10.1093/comjnl/7.4.308.

[93] H. M. Nyew, O. Abdelkhalik, and N. Onder. "Structured-Chromosome Evolutionary Algorithms for Variable-Size Autonomous Interplanetary Trajectory Planning Optimization". In: *Journal of Aerospace Information Systems* 12.3 (2015), pp. 314–328. ISSN: 2327-3097. DOI: 10.2514/1.I010272.

[94] M. A. Oliver and R. Webster. "Kriging: a method of interpolation for geographical information systems". In: *International Journal of Geographical Information Systems* 4.3 (1990), pp. 313–332. ISSN: 0269-3798. DOI: 10.1080/02693799008941549.

[95] M. Papadrakakis, N. D. Lagaros, and Y. Tsompanakis. "Structural optimization using evolution strategies and neural networks". In: *Computer methods in applied mechanics and engineering* 156.1-4 (1998), pp. 309–333.

[96] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. "Efficient global optimization of constrained mixed variable problems". In: *Journal of Global Optimization* 73.3 (2019), pp. 583–613.

[97] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. "How to Deal with Mixed-Variable Optimization Problems: An Overview of Algorithms and Formulations". In: *Advances in Structural and Multidisciplinary Optimization*. Cham: Springer International Publishing, 2018, pp. 64–82. DOI: 10.1007/978-3-319-67988-4_5.

[98] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. "Overview and Comparison of Gaussian Process-Based Surrogate Models for Mixed Continuous and Discrete Variables: Application on Aerospace Design Problems". In: *High-Performance Simulation-Based Optimization*. Springer, 2020, pp. 189–224.

[99] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. "Surrogate model based optimization of constrained mixed variable problems: application to the design of a launch vehicle thrust frame". In: *AIAA Scitech 2019 Forum*. 2019, p. 1971.

[100] V. Picheny, T. Wagner, and D. Ginsbourger. "A benchmark of kriging-based infill criteria for noisy optimization". In: *Structural and Multidisciplinary Optimization* 48.3 (2013), pp. 607–626.

[101] J. Pinheiro and D. M. Bates. "Unconstrained parametrizations for variance-covariance matrices". In: *Statistics and Computing* 6.3 (1996), pp. 289–296. ISSN: 0960-3174. DOI: 10.1007/BF00140873.

[102] N Prasadh, R Moss, K Collett, A. Nelessen, S Edwards, and D. Mavris. "A systematic method for sme-driven space system architecture down-selection". In: *AIAA SPACE 2014 conference and exposition*. 2014, pp. 4–7.

[103] R. Priem, N. Bartoli, and Y. Diouane. "On the Use of Upper Trust Bounds in Constrained Bayesian Optimization Infill Criteria". In: *AIAA Aviation 2019 Forum*. 2019, p. 2986.

[104] P. Z. G. Qian, H. Wu, and C. F. J. Wu. "Gaussian Process Models for Computer Experiments With Qualitative and Quantitative Factors". In: *Technometrics* 50.3 (2008), pp. 383–396. ISSN: 0040-1706. DOI: 10.1198/004017008000000262.

[105] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker. "Surrogate-based analysis and optimization". In: *Progress in Aerospace Sciences* 41.1 (2005), pp. 1–28. ISSN: 03760421. DOI: 10.1016/j.paerosci.2005.02.001.

[106] K. Rashid, S. Ambani, and E. Cetinkaya. "An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization". In: *Engineering Optimization* 45.2 (2008), pp. 185–206. ISSN: 0305215X. DOI: 10.1080/0305215X.2012.665450.

[107] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006, p. 248. ISBN: 026218253X.

[108] O. Roustant, E. Padonou, Y. Deville, A. Clément, G. Perrin, J. Giorla, and H. Wynn. "Group kernels for Gaussian process metamodels with categorical inputs". In: *arXiv preprint arXiv:1802.02368* (2018).

[109] S. Roy, W. A. Crossley, B. Stanford, K. T. Moore, and J. S. Gray. "A Mixed Integer Efficient Global Optimization Algorithm with Multiple Infill Strategy-Applied to a Wing Topology Optimization Problem". In: *AIAA Scitech 2019 Forum*. 2019, p. 2356.

[110] S. Roy, K. Moore, J. T. Hwang, J. S. Gray, W. A. Crossley, and J. Martins. "A Mixed Integer Efficient Global Optimization Algorithm for the Simultaneous Aircraft Allocation-Mission-Design Problem". In: *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, 2017. ISBN: 978-1-62410-453-4. DOI: 10.2514/6.2017-1305.

[111] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. "Design and Analysis of Computer Experiments". In: *Statistical Science* 4.4 (1989), pp. 409–423. ISSN: 0883-4237. DOI: 10.1214/ss/1177012413.

[112] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer New York, 2003, p. 286. ISBN: 1475737998.

[113] W. S. Sarle. "Neural networks and statistical models". In: (1994).

[114] M. J. Sasena. "Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations". PhD thesis. 2002, pp. 1–237. DOI: 10.1.1.2.4697.

[115] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[116] M. Schonlau, W. J. Welch, and D. R. Jones. "Global versus local search in constrained optimization of computer models". In: *Lecture Notes-Monograph Series* (1998), pp. 11–25.

[117] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen. "Metamodels for Computer-based Engineering Design: Survey and recommendations". In: *Engineering with Computers* 17.2 (2001), pp. 129–150. ISSN: 0177-0667. DOI: 10.1007/PL00007198. arXiv: arXiv:1011.1669v3.

[118] A. J. Smola and B. Schölkopf. "A tutorial on support vector regression". In: *Statistics and computing* 14.3 (2004), pp. 199–222.

[119] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[120] M. Stelmack and S. Batill. "Neural network approximation of mixed continuous/discrete systems in multidiscplinary design". In: *36th AIAA Aerospace Sciences Meeting and Exhibit*. 1998, p. 916.

[121] M. Stelmack, N. Nakashima, and S. Batill. "Genetic algorithms for mixed discrete/continuous optimization in multidisciplinary design". In: *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Reston, Virigina: American Institute of Aeronautics and Astronautics, 1998. DOI: 10.2514/6.1998-4771.

[122] R. Storn and K. Price. "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". In: *Journal of Global Optimization* 11.4 (1997), pp. 341–359. ISSN: 09255001. DOI: 10.1023/A:1008202821328.

[123] D. B. Suits. "Use of Dummy Variables in Regression Equations". In: *Journal of the American Statistical Association* 52.280 (1957), pp. 548–551. ISSN: 1537274X. DOI: 10.1080/01621459.1957.10501412. arXiv: arXiv:1011.1669v3.

[124] C. Sun, J. Zeng, and J.-S. Pan. "A modified particle swarm optimization with feasibility-based rules for mixed-variable optimization problems". In: *International Journal of Innovative Computing, Information and Control* 7.6 (2011), pp. 3081–3096.

[125] L. P. Swiler, P. D. Hough, P. Qian, X. Xu, C. Storlie, and H. Lee. "Surrogate models for mixed discrete-continuous variables". In: *Constraint Programming and Decision Making*. Springer, 2014, pp. 181–202.

[126] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009, p. 593. ISBN: 9780470278581.

[127] G. Venter and J. Sobieszczanski-Sobieski. "Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization". In: *Structural and Multidisciplinary Optimization* 26.1-2 (2004), pp. 121–131. ISSN: 1615-147X. DOI: 10.1007/s00158-003-0318-3.

[128] G. G. Wang and S. Shan. "Review of Metamodeling Techniques in Support of Engineering Design Optimization". In: *Journal of Mechanical Design* 129.4 (2007), p. 370. ISSN: 10500472. DOI: 10.1115/1.2429697.

[129] J. Wang and Z. Yin. "A ranking selection-based particle swarm optimizer for engineering design optimization problems". In: *Structural and Multidisciplinary Optimization* 37.2 (2008), pp. 131–147. ISSN: 1615-147X. DOI: 10.1007/s00158-007-0222-3.

[130] J. R. Wertz. "Mission geometry: orbit and constellation design and management: spacecraft orbit and attitude systems". In: *Mission geometry: orbit and constellation design and management: spacecraft orbit and attitude systems/James R. Wertz. El Segundo, CA; Boston: Microcosm: Kluwer Academic Publishers, 2001. Space technology library; 13* (2001).

[131] H. White. *Artificial neural networks: approximation and learning theory*. Blackwell Publishers, Inc., 1992.

[132] M. Zaefferer, J. Stork, and T. Bartz-Beielstein. "Distance measures for permutations in combinatorial efficient global optimization". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2014, pp. 373–383.

[133] M. Zaefferer, J. Stork, M. Friese, A. Fischbach, B. Naujoks, and T. Bartz-Beielstein. "Efficient global optimization for combinatorial problems". In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM. 2014, pp. 871–878.

[134] Y. Zhang, D. Apley, and W. Chen. "Bayesian Optimization for Materials Design with Mixed Quantitative and Qualitative Variables". In: *arXiv preprint arXiv:1910.01688* (2019).

[135] Y. Zhang, S. Tao, W. Chen, and D. W. Apley. "A latent variable approach to Gaussian process modeling with qualitative and quantitative factors". In: *Technometrics* (2019), pp. 1–12.

[136] Y. Zhang and W. I. Notz. "Computer experiments with qualitative and quantitative variables: A review and reexamination". In: *Quality Engineering*. Vol. 27. 1. Taylor & Francis, 2015, pp. 2–13. DOI: 10.1080/08982112.2015.968039.

[137] Q. Zhou, P. Z. G. Qian, and S. Zhou. "A Simple Approach to Emulation for Computer Models With Qualitative and Quantitative Factors". In: *Technometrics* 53.3 (2011), pp. 266–273. ISSN: 0040-1706. DOI: 10.1198/TECH.2011.10025.

APPENDIX A

Publications and communications

Articles

- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2019). *Efficient global optimization of constrained mixed variable problems*, Journal of Global Optimization, 73(3), 583-613.
- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2020). *Bayesian Optimization of variable-size design space problems.* Optimization and Engineering, *submitted.*

Chapters

- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2020). *Overview and Comparison of Gaussian Process-Based Surrogate Models for Mixed Continuous and Discrete Variables: Application on Aerospace Design Problems.* In High-Performance Simulation-Based Optimization (pp. 189-224). Springer, Cham.
- Brevault, L., **Pelamatti, J.**, Balesdent, M., Talbi, E. G., & Melab, N. (2020). *MDO related issues: multi-objective and mixed continuous / discrete optimization.* In Aerospace System Analysis and Optimization in Uncertainty, Springer Nature Switzerland AG.

Communications

- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2017). *How to deal with mixed-variable optimization problems: An overview of algorithms and formulations.* In 12th World Congress of Structural and Multidisciplinary Optimization. 2017 Braunschweig, Germany. Springer, Cham.
- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2019). *Surrogate model based optimization of constrained mixed variable problems.* In 8th International Conference on Bioinspired Optimization Methods and their Application. 2018 Paris, France.
- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2019). *Surrogate model based optimization of constrained mixed variable problems: application to the design of a launch vehicle thrust frame.* In AIAA Scitech Forum. 2019, San Diego, USA.

- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2019) *Surrogate model-based strategy for variable-size design space optimization problems.* In 13th World Congress of Structural and Multidisciplinary Optimization. 2019 Beijing, China, *publication pending.*
- **Pelamatti, J.**, Brevault, L., Balesdent, M., Talbi, E. G., & Guerin, Y. (2019) *Grouped hierarchical kernel for the Efficient Global Optimization of variable-size design space problems.* In 13th World Congress of Structural and Multidisciplinary Optimization. 2019 Beijing, China, *publication pending.*

APPENDIX B

Proof of validity of the mixed-variable noise handling kernel

In order to take into account the possible presence of noisy data, it is common practice to add a noise handling term to the global kernel defined as:

$$\sigma_n^2 \delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) \tag{B.1}$$

where σ_n^2 is a noise handling hyperparameter (usually proportional to the noise magnitude) and δ_n is a kernel function similar to the Kronecker delta defined as:

$$\delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) = \begin{cases} 1 & \text{if } \{\mathbf{x}, \mathbf{z}\} = \{\mathbf{x}', \mathbf{z}'\} \\ 0 & \text{if } \{\mathbf{x}, \mathbf{z}\} \neq \{\mathbf{x}', \mathbf{z}'\} \end{cases} \tag{B.2}$$

The validity of this approach is ensured under the condition that $\delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\})$ is a valid kernel on the mixed-variable design space. Similarly to what is done in Chapter 3, $\delta_n(\cdot)$ can be decomposed into a product of a purely continuous kernel and a purely discrete kernel:

$$\delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\}) = \delta_{n_x}(\mathbf{x}, \mathbf{x}')\delta_{n_z}(\mathbf{z}, \mathbf{z}') \tag{B.3}$$

where:

$$\delta_{n_x}(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}' \\ 0 & \text{if } \mathbf{x} \neq \mathbf{x}' \end{cases} \tag{B.4}$$

and

$$\delta_{n_z}(\mathbf{z}, \mathbf{z}') = \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{z}' \\ 0 & \text{if } \mathbf{z} \neq \mathbf{z}' \end{cases} \tag{B.5}$$

The valid construction of $\delta_{n_x}(\cdot)$ can be derived by representing it as a squared exponential kernel and by considering a limit value of the length scale hyperparamter θ which tends to infinity:

$$\delta_{n_x}(\mathbf{x}, \mathbf{x}') = \lim_{\theta \to \infty} \exp(-\theta ||\mathbf{x} - \mathbf{x}'||^2) \tag{B.6}$$

The kernel defined above returns a value of 1 only if $\mathbf{x}$ and $\mathbf{x}'$ are identical, and 0 otherwise.

In order to validate the construction of $\delta_{n_z}(\cdot)$, a similar approach as for the CS kernel presented in Section 3.5.1 can be followed. More specifically, a one-hot encoding of the discrete variable vector $\mathbf{z}$ is considered. Let $\mathbf{z}$ be characterized by m categories and let $\phi(\cdot)$ be a mapping of the discrete input space onto a m-dimensional Hilbert space: $\phi(\mathbf{z}) : F_z \to \mathbb{R}^m$. The mapping is defined in such a way that the only non-zero coordinate of the image in the Hilbert space corresponds to

the dimension associated to the mapped category. An example of the mapping described above for a generic discrete variable characterized by 4 categories is provided below:

$$\mathbf{z} \in \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4\} \rightarrow \begin{cases} \phi(\mathbf{z} = \mathbf{z}_1) = [1,0,0,0] \\ \phi(\mathbf{z} = \mathbf{z}_2) = [0,1,0,0] \\ \phi(\mathbf{z} = \mathbf{z}_3) = [0,0,1,0] \\ \phi(\mathbf{z} = \mathbf{z}_4) = [0,0,0,1] \end{cases}$$

A valid construction of $\delta_{n_z}(\cdot)$ can then be obtained by defining the inner product on the m-dimensional Hilbert space as the scalar product:

$$\delta_{n_z}(\mathbf{z}, \mathbf{z}') = \langle \phi(\mathbf{z}), \phi(\mathbf{z}') \rangle = \phi(\mathbf{z}) \cdot \phi(\mathbf{z}') \tag{B.7}$$

which returns a value of 1 only if $\mathbf{z}$ and $\mathbf{z}'$ belong to the same category, and 0 otherwise.

It is therefore shown that $\delta_n(\{\mathbf{x}, \mathbf{z}\}, \{\mathbf{x}', \mathbf{z}'\})$ is valid by construction. By consequence, this noisy data handling approach can be extended to the mixed-variable case without requiring additional adaptations.

APPENDIX C

Variable-size design space Goldstein function

The variable-size design space variant of the Goldstein function which is considered for the testing discussed in Chapter 5 is characterized by a global design space with 5 continuous design variables, 4 discrete design variables and 2 dimensional design variables. Depending on the dimensional variable values, 8 different sub-problems can be identified, with total dimensions of 6 or 7, ranging from 2 continuous variables and 4 discrete variables to 5 continuous variables and 2 discrete variables. All of the sub-problem are subject to a variable-size design space constraint.

The resulting optimization problem can be defined as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x},\mathbf{z},\mathbf{w}) \\ \text{w.r.t.} \quad & \mathbf{x} = \{x_1,\dots,x_5\} \ \text{ with } \ x_i \in [0,100] \ \text{ for } i = 1,5 \\ & \mathbf{z} = \{z_1,\dots,z_4\} \ \text{ with } \ z_i \in \{0,1,2\} \ \text{ for } i = 1,4 \\ & \mathbf{w} = \{w_1,w_2\} \ \text{ with } \ w_1 \in \{0,1,2,3\} \ \text{ and } w_2 \in \{0,1\} \\ \text{s.t.:} \quad & g(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0 \end{aligned} \tag{C.1}$$

where:

$$f(\mathbf{x},\mathbf{z},\mathbf{w}) = \begin{cases} f_1(x_1,x_2,z_1,z_2,z_3,z_4) & \text{if } w_1 = 0 \text{ and } w_2 = 0 \\ f_2(x_1,x_2,x_3,z_2,z_3,z_4) & \text{if } w_1 = 1 \text{ and } w_2 = 0 \\ f_3(x_1,x_2,x_4,z_1,z_3,z_4) & \text{if } w_1 = 2 \text{ and } w_2 = 0 \\ f_4(x_1,x_2,x_3,x_4,z_3,z_4) & \text{if } w_1 = 3 \text{ and } w_2 = 0 \\ f_5(x_1,x_2,x_5,z_1,z_2,z_3,z_4) & \text{if } w_1 = 0 \text{ and } w_2 = 1 \\ f_6(x_1,x_2,x_3,x_5,z_2,z_3,z_4) & \text{if } w_1 = 1 \text{ and } w_2 = 1 \\ f_7(x_1,x_2,x_4,x_5,z_1,z_3,z_4) & \text{if } w_1 = 2 \text{ and } w_2 = 1 \\ f_8(x_1,x_2,x_3,x_5,x_4,z_3,z_4) & \text{if } w_1 = 3 \text{ and } w_2 = 1 \end{cases} \tag{C.2}$$

and:

$$g(\mathbf{x},\mathbf{z},\mathbf{w}) = \begin{cases} g_1(x_1,x_2,z_1,z_2) & \text{if } w_1 = 0 \\ g_2(x_1,x_2,z_2) & \text{if } w_1 = 1 \\ g_3(x_1,x_2,z_1) & \text{if } w_1 = 2 \\ g_4(x_1,x_2,z_3,z_4) & \text{if } w_1 = 3 \end{cases} \tag{C.3}$$

The objective functions $f_1(\cdot),\dots,f_8(\cdot)$ are defined as follows:

$$
\begin{aligned}
f_1(x_1, x_2, z_1, z_2, z_3, z_4) = \quad & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\
& 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\
& 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\
& 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\
& 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\
& 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\
& 5.62199x_2x_3x_4 \cdot 10^{-5}
\end{aligned} \tag{C.4}
$$

where x_3 and x_4 are defined as a function of z_1 and z_2 according to the relations defined in Table C.1.

	$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$z_2 = 0$	$x_3 = 20, x_4 = 20$	$x_3 = 50, x_4 = 20$	$x_3 = 80, x_4 = 20$
$z_2 = 1$	$x_3 = 20, x_4 = 50$	$x_3 = 50, x_4 = 50$	$x_3 = 80, x_4 = 50$
$z_2 = 2$	$x_3 = 20, x_4 = 80$	$x_3 = 50, x_4 = 80$	$x_3 = 80, x_4 = 80$

TABLE C.1: Characterization of the variable-dimension search space Goldstein function sub-problem N°1 discrete categories

$$
\begin{aligned}
f_2(x_1, x_2, x_3, z_2, z_3, z_4) = \quad & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\
& 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\
& 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\
& 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\
& 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\
& 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\
& 5.62199x_2x_3x_4 \cdot 10^{-5}
\end{aligned} \tag{C.5}
$$

where x_4 is defined as a function of z_2 according to the relations defined in Table C.2.

$z_2 = 0$	$z_2 = 1$	$z_2 = 2$
$x_4 = 20$	$x_4 = 50$	$x_4 = 80$

TABLE C.2: Characterization of the variable-dimension search space Goldstein function sub-problem N°2 discrete categories

$$
\begin{aligned}
f_3(x_1, x_2, x_4, z_1, z_3, z_4) = \quad & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\
& 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\
& 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\
& 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\
& 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\
& 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\
& 5.62199x_2x_3x_4 \cdot 10^{-5}
\end{aligned} \tag{C.6}
$$

where x_3 is defined as a function of z_1 according to the relations defined in Table C.3.

$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$x_3 = 20$	$x_3 = 50$	$x_3 = 80$

TABLE C.3: Characterization of the variable-dimension search space Goldstein function sub-problem N°2 discrete categories

$$\begin{aligned} f_4(x_1, x_2, x_3, x_4, z_3, z_4 = \quad & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\ & 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\ & 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\ & 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\ & 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\ & 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\ & 5.62199x_2x_3x_4 \cdot 10^{-5} \end{aligned} \tag{C.7}$$

$$\begin{aligned} f_5(x_1, x_2, z_1, z_2, z_3, z_4) = \quad & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\ & 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\ & 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\ & 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\ & 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\ & 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\ & 5.62199x_2x_3x_4 \cdot 10^{-5} + 5\cos(2\pi\frac{x_5}{100}) - 2 \end{aligned} \tag{C.8}$$

where x_3 and x_4 are defined as a function of z_1 and z_2 according to the relations defined in Table C.4.

	$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$z_2 = 0$	$x_3 = 20, x_4 = 20$	$x_3 = 50, x_4 = 20$	$x_3 = 80, x_4 = 20$
$z_2 = 1$	$x_3 = 20, x_4 = 50$	$x_3 = 50, x_4 = 50$	$x_3 = 80, x_4 = 50$
$z_2 = 2$	$x_3 = 20, x_4 = 80$	$x_3 = 50, x_4 = 80$	$x_3 = 80, x_4 = 80$

TABLE C.4: Characterization of the variable-dimension search space Goldstein function sub-problem N°5 discrete categories

$$\begin{aligned} f_6(x_1, x_2, x_3, z_2, z_3, z_4) = \quad & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\ & 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\ & 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\ & 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\ & 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\ & 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\ & 5.62199x_2x_3x_4 \cdot 10^{-5} + 5\cos(2\pi\frac{x_5}{100}) - 2 \end{aligned} \tag{C.9}$$

where x_4 is defined as a function of z_2 according to the relations defined in Table C.5.

$z_2 = 0$	$z_2 = 1$	$z_2 = 2$
$x_4 = 20$	$x_4 = 50$	$x_4 = 80$

TABLE C.5: Characterization of the variable-dimension search space Goldstein function sub-problem N°6 discrete categories

$$\begin{aligned} f_7(x_1, x_2, x_4, z_1, z_3, z_4) = \; & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\ & 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\ & 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\ & 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\ & 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\ & 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\ & 5.62199x_2x_3x_4 \cdot 10^{-5} + 5\cos(2\pi\frac{x_5}{100}) - 2 \end{aligned} \tag{C.10}$$

where x_3 is defined as a function of z_1 according to the relations defined in Table C.6.

$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$x_3 = 20$	$x_3 = 50$	$x_3 = 80$

TABLE C.6: Characterization of the variable-dimension search space Goldstein function sub-problem N°7 discrete categories

$$\begin{aligned} f_8(x_1, x_2, x_3, x_4, z_3, z_4 = \; & 53.3108 + 0.184901x_1 - 5.02914x_1^3 \cdot 10^{-6} + 7.72522x_1^{z_3} \cdot 10^{-8} - \\ & 0.0870775x_2 - 0.106959x_3 + 7.98772x_3^{z_4} \cdot 10^{-6} + \\ & 0.00242482x_4 + 1.32851x_4^3 \cdot 10^{-6} - 0.00146393x_1x_2 - \\ & 0.00301588x_1x_3 - 0.00272291x_1x_4 + 0.0017004x_2x_3 + \\ & 0.0038428x_2x_4 - 0.000198969x_3x_4 + 1.86025x_1x_2x_3 \cdot 10^{-5} - \\ & 1.88719x_1x_2x_4 \cdot 10^{-6} + 2.50923x_1x_3x_4 \cdot 10^{-5} - \\ & 5.62199x_2x_3x_4 \cdot 10^{-5} + 5\cos(2\pi\frac{x_5}{100}) - 2 \end{aligned} \tag{C.11}$$

The constraints $g_1(\cdot), \ldots, g_4(\cdot)$ are defined are defined as follows:

$$g_1(x_1, x_2, z_1, z_2) = -(x_1 - 50)^2 - (x_2 - 50)^2 + (20 + c_1 * c_2)^2 \tag{C.12}$$

where c_1 and c_2 are defined as a function of z_1 and z_2 according to the relations defined in Table C.7.

$$g_2(x_1, x_2, z_2) = -(x_1 - 50)^2 - (x_2 - 50)^2 + (20 + c_1 * c_2)^2 \tag{C.13}$$

where $c_1 = 0.5$ and c_2 is defined as a function of z_2 according to the relations defined in Table C.8.

$$g_3(x_1, x_2, z_1) = -(x_1 - 50)^2 - (x_2 - 50)^2 + (20 + c_1 * c_2)^2 \tag{C.14}$$

	$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$z_2 = 0$	$c_1 = 3, c_2 = 0.5$	$c_1 = 2, c_2 = 0.5$	$c_1 = 1, c_2 = 0.5$
$z_2 = 1$	$c_1 = 3, c_2 = -1$	$c_1 = 2, c_2 = -1$	$c_1 = 1, c_2 = -1$
$z_2 = 1$	$c_1 = 3, c_2 = -2$	$c_1 = 2, c_2 = -2$	$c_1 = 1, c_2 = -2$

TABLE C.7: Characterization of the variable-dimension search space Goldstein function constraint

$z_2 = 0$	$z_2 = 1$	$z_2 = 2$
$c_2 = 0.5$	$c_2 = -1$	$c_2 = -2$

TABLE C.8: Characterization of the variable-dimension search space Goldstein function constraint

where $c_2 = 0.7$ and c_1 is defined as a function of z_1 according to the relations defined in Table C.9.

$z_1 = 0$	$z_1 = 1$	$z_1 = 2$
$c_1 = 3$	$c_1 = 2$	$c_1 = 1$

TABLE C.9: Characterization of the variable-dimension search space Goldstein function constraint

$$g_4(x_1, x_2, z_3, z_4) = -(x_1 - 50)^2 - (x_2 - 50)^2 + (20 + c_1 * c_2)^2 \tag{C.15}$$

where c_1 and c_2 are defined as a function of z_3 and z_4 according to the relations defined in Table C.10.

	$z_3 = 0$	$z_3 = 1$	$z_3 = 2$
$z_4 = 0$	$c_1 = 3, c_2 = 0.5$	$c_1 = 2, c_2 = 0.5$	$c_1 = 1, c_2 = 0.5$
$z_4 = 1$	$c_1 = 3, c_2 = -1$	$c_1 = 2, c_2 = -1$	$c_1 = 1, c_4 = -1$
$z_4 = 2$	$c_1 = 3, c_2 = -2$	$c_1 = 2, c_2 = -2$	$c_1 = 1, c_2 = -2$

TABLE C.10: Characterization of the variable-dimension search space Goldstein function constraint

Résumé

Dans le cadre de la conception de systèmes complexes, tels que les aéronefs et les lanceurs, la présence de fonctions d'objectifs et/ou de contraintes à forte intensité de calcul (*e.g.,* modèles d'éléments finis et analyses multidisciplinaires) couplée à la dépendance de choix de conception technologique discrets et non ordonnés entraîne des problèmes d'optimisation difficiles. De plus, une partie de ces choix technologiques est associée à un certain nombre de variables de conception continues et discrètes spécifiques qui ne doivent être prises en considération que si des choix technologiques et/ou architecturaux spécifiques sont faits. Par conséquent, le problème d'optimisation qui doit être résolu afin de déterminer la conception optimale du système présente un espace de recherche et un domaine de faisabilité variant de fa con dynamique.

Les quelques algorithmes existants qui permettent de résoudre ce type particulier de problèmes ont tendance à exiger une grande quantité d'évaluations de fonctions afin de converger vers l'optimum réalisable, et sont donc inadéquats lorsqu'il s'agit de résoudre des problèmes à forte intensité de calcul qui peuvent souvent être rencontrés dans le cadre de la conception de systèmes complexes. Pour cette raison, cette thèse explore la possibilité de résoudre des problèmes à espace de conception contraint à variables mixtes et de taille variable. Les approches développées s'appuient sur des méthodes d'optimisation à base de modèles de substitution créés à l'aide de processus Gaussiens, également connues sous le nom d'optimisation Bayésienne. Plus spécifiquement, 3 axes principaux sont discutés. Premièrement, la modélisation de substitution par processus Gaussien de fonctions mixtes continues/discrètes et les défis qui y sont associés sont discutés en détail. Un formalisme unificateur est proposé afin de faciliter la description et la comparaison entre les noyaux existants permettant d'adapter les processus Gaussiens à la présence de variables discrètes non ordonnées. De plus, les performances de modélisation de ces différents noyaux sont testées et comparées sur un ensemble de benchmarks analytiques et de conception ayant des caractéristiques et des paramétrages différents.

Dans la deuxième partie de la thèse, la possibilité d'étendre la modélisation de substitution mixte continue/discrète à un contexte d'optimisation Bayésienne est discutée. La faisabilité théorique de cette extension en termes de modélisation des fonctions objectif et de contrainte ainsi que de définition et d'optimisation de la fonction d'acquisition est démontrée. Différentes alternatives possibles sont considérées et décrites. Enfin, la performance de l'algorithme d'optimisation proposé, avec diverses paramétrisations des noyaux et différentes initialisations, est testée sur un certain nombre de cas-test analytiques et de conception et est comparée aux algorithmes de référence.

Dans la dernière partie de ce manuscrit, deux approches permettant d'adapter les algorithmes d'optimisation Bayésienne mixte continue/discrète discutés précédemment afin de résoudre des problèmes caractérisés par un espace de conception de taille variable (*i.e.,* variant dynamiquement au cours de l'optimisation) sont proposées. La première adaptation est basée sur l'optimisation parallèle de plusieurs sous-problèmes couplée à une allocation de budget de calcul basée sur l'information fournie par les modèles de substitution. La seconde adaptation, au contraire, est basée sur la définition d'un noyau permettant de calculer la covariance entre des échantillons appartenant à des espaces de recherche partiellement différents en fonction du regroupement hiérarchique des variables dimensionnelles. Enfin, les deux alternatives sont testées et comparées sur un ensemble de cas-test analytiques et de conception.

Globalement, il est démontré que les méthodes d'optimisation proposées permettent de converger vers les optima des différents types de problèmes contraints considérablement plus rapidement par rapport aux méthodes existantes. Elles représentent donc un outil prometteur pour la conception de systèmes complexes.

Abstract

Within the framework of complex system design, such as aircraft and launch vehicles, the presence of computationally intensive objective and/or constraint functions (*e.g.,* finite element models and multidisciplinary analyses) coupled with the dependence on discrete and unordered technological design choices results in challenging optimization problems. Furthermore, part of these technological choices is associated to a number of specific continuous and discrete design variables which must be taken into consideration only if specific technological and/or architectural choices are made. As a result, the optimization problem which must be solved in order to determine the optimal system design presents a dynamically varying search space and feasibility domain.

The few existing algorithms which allow solving this particular type of problems tend to require a large amount of function evaluations in order to converge to the feasible optimum, and result therefore inadequate when dealing with the computationally intensive problems which can often be encountered within the design of complex systems. For this reason, this thesis explores the possibility of performing constrained mixed-variable and variable-size design space optimization by relying on surrogate model-based design optimization performed with the help of Gaussian processes, also known as Bayesian optimization. More specifically, 3 main axes are discussed. First, the Gaussian process surrogate modeling of mixed continuous/discrete functions and the associated challenges are extensively discussed. A unifying formalism is proposed in order to facilitate the description and comparison between the existing kernels allowing to adapt Gaussian processes to the presence of discrete unordered variables. Furthermore, the actual modeling performances of these various kernels are tested and compared on a set of analytical and design related benchmarks with different characteristics and parameterizations.

In the second part of the thesis, the possibility of extending the mixed continuous/discrete surrogate modeling to a context of Bayesian optimization is discussed. The theoretical feasibility of said extension in terms of objective/-constraint function modeling as well as acquisition function definition and optimization is shown. Different possible alternatives are considered and described. Finally, the performance of the proposed optimization algorithm, with various kernels parameterizations and different initializations, is tested on a number of analytical and design related test-cases and compared to reference algorithms.

In the last part of this manuscript, two alternative ways of adapting the previously discussed mixed continuous/discrete Bayesian optimization algorithms in order to solve variable-size design space problems (*i.e.,* problems characterized by a dynamically varying design space) are proposed. The first adaptation is based on the parallel optimization of several sub-problems coupled with a computational budget allocation based on the information provided by the surrogate models. The second adaptation, instead, is based on the definition of a kernel allowing to compute the covariance between samples belonging to partially different search spaces based on the hierarchical grouping of design variables. Finally, the two alternatives are tested and compared on a set of analytical and design related benchmarks.

Overall, it is shown that the proposed optimization methods allow to converge to the various constrained problem optimum neighborhoods considerably faster when compared to the reference methods, thus representing a promising tool for the design of complex systems.